Active Topology Inference using Network Coding

Pegah Sattari, *IEEE Student Member*, Christina Fragouli, *IEEE Member*, and Athina Markopoulou, *IEEE Member*

***Abstract*—Our goal, in this paper, is to infer the topology of a network when (i) we can send probes between sources and receivers at the edge of the network and (ii) intermediate nodes can perform simple network coding operations, *i.e.*, additions. Our key intuition is that network coding introduces topology-dependent correlation in the observations at the receivers, which can be exploited to infer the topology. For tree topologies, we design hierarchical clustering algorithms, building on our prior work in [1]. For directed acyclic graphs (DAGs), first we decompose the topology into a number of two source, two receiver subnetwork components and then we merge these components to reconstruct the topology. Our approach for DAGs builds on prior work on tomography [2], and improves upon it by employing network coding to accurately distinguish among all different 2-by-2 components. We evaluate our algorithms through simulation of a number of realistic topologies.**

***Index Terms*—Network Coding, Topology Inference.**

I. Introduction

Knowledge of network topology is important for network management, diagnosis, operations, security and performance optimization. Depending on the context, “topology” may refer to different layers, such as the Internet’s router-level topology, an overlay network or a peer-to-peer topology, a wireless ad-hoc network topology etc. Topology may not be available for various reasons, *e.g.,* either because operators do not want to reveal the internal characteristics of their network to the outside world, and/or because topology changes frequently.

Due to the importance of network topology, a large body of prior work has focused on measurement of network topology. A family of techniques are based on `traceroute`-like measurements, which collect the ids of nodes across `traceroute` paths and use them to reconstruct the topology. Another family of techniques are tomographic: network tomography aims at inferring internal network characteristics, including topology, using end-to-end probes, putting the processing burden on a few end-nodes and keeping internal nodes simple. More specifically, multicast or unicast probes are sent and received between sets of nodes and the topology is inferred based on the number and order of received probes. In this paper, we revisit the problem of topology inference using end-to-end probes, but in a network with network coding capabilities.

The network coding paradigm is based on the idea that intermediate nodes linearly combine packets and receivers solve a system of linear equations to recover the original packets. This idea has been shown to bring benefits, for example in terms of throughput, complexity and reliability. The network coding idea is well-matched to content distribution over peer-to-peer networks, overlay networks, wireless multihop, etc. In this paper, we consider networks that are already equipped with simple network coding capabilities and we design active probing schemes that make use of these capabilities to improve topology inference.

Our key intuition is that network coding introduces topology-dependent correlation in the content of packets observed at the receivers, which can then be exploited to reverse-engineer the topology. For example, a coding point, *i.e.*, a node that combines one or more incoming packets, introduces correlation between packets coming from different sources, in a similar way that multicast introduces correlation in the packets sent by the same source and received by several receivers. In fact, this correlation introduced by multicast, has been the starting point and the main idea underlying tomographic topology inference. Subsequent schemes made this idea more practical, by emulating multicast with back-to-back unicast probes [2], [3]. In contrast, relating probes from different sources, in order to reveal intermediate nodes, has been a challenge in the tomography literature. Using simple network coding operations at coding points solves this problem and allows accurate and fast topology inference.

First, we consider undirected trees, where leaves can act as both sources and receivers of probe packets, and we design hierarchical clustering algorithms that infer the topology, building on our prior work in [1]. Then, we consider general graphs, in particular directed acyclic graphs (DAGs) with a fixed set of M sources and N receivers and a pre-determined routing scheme. We first decompose the topology into a number of two source, two receiver subnetwork components and then we merge these components to reconstruct the topology. Our approach for DAGs builds on prior work on tomography [2], but improves upon it by employing simple network coding operations at intermediate nodes (just additions) to deterministically distinguish among all candidate 2-by-2 subnetwork components, which was impossible without network coding [2], [3]. We evaluate our algorithms through simulation over a number of topologies and we show that they can infer the topology accurately and faster than traditional approaches.

The structure of the rest of the paper is as follows. Section II discusses related work. Section III presents our assumptions, notation and problem statement. Section IV presents algorithms for inferring tree topologies. Binary trees are discussed in Section IV-A, in the absence (Section IV-A1) or presence (Section IV-A2) of packet loss. General trees are discussed in Sections IV-B1 and IV-B2. Section V presents algorithms for inferring general directed acyclic graphs (DAGs). Section V-B presents our algorithms for inferring 2-by-2 subnetwork components, in the absence (Section V-B1) or presence (Section V-B2) of packet loss. Section V-C explains how to merge

P. Sattari and A. Markopoulou are with the EECS Department at the University of California, Irvine, CA, 92697 USA. E-mail: {*psattari, athina*}*@uci.edu*.

C. Fragouli is with the School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland, Email: *christina.fragouli@epfl.ch*.

these components to reconstruct the entire topology. Section VI provides simulation results for random graphs as well as real Internet topologies. Section VII provides an in-depth comparison and makes connections between our approach and alternative topology inference approaches. Section VIII concludes the paper.

II. Related Work

There are two bodies of related work: one from the network tomography and another from the network coding literature.

A good survey of network tomography can be found in [4]. In our work, we are interested in inferring the topology based on end-to-end probes, rather than link-level characteristics. An early work on topology inference using end-to-end measurements is [5], where the correlation between end-to-end multicast packet loss patterns was used to infer the topology of binary trees. The correctness of this idea was rigorously established in [6], and was extended to general trees and to measurements other than loss, such as delay variance [7], or more generally any metric that grows monotonically with the number of traversed links. The idea has also been extended to unicast probes [3], [8]. In summary, tomographic schemes for topology inference use end-to-end active probing and feed the number, order or a monotonic property (*e.g.,* delay variance or loss) of received probes as input to statistical signal-processing techniques. Inference of link characteristics [9] can also be combined with topology inference [3].

Most tomographic techniques rely on probes sent from a single source in a tree topology [5]–[8], [10]–[17]. The work by Rabbat et al. introduced the multiple-source multiple-destination (M-by-N) tomography problem, *i.e.,* sending probes between M sources and N destinations [2], [3]. They showed that an M-by-N topology can be decomposed into a collection of 2-by-2 components. They proposed to coordinate transmission of multi-packet probes from two sources and to measure the packet arrival order at the two receivers in order to infer some information about the 2-by-2 topology. Assuming knowledge of M 1-by-N topologies and of all 2-by-2 components, they also showed how to merge a second source's 1-by-N topology with the first. The resulting M-by-N topology is not exact, but provides bounds on the locations of joining points with respect to the branching points. It also requires a large number of probes, as do all approaches that need to collect enough probes for statistical significance [7], [8], [12], [15], [17], [18]. Our work on DAGs builds on and extends the multiple-source multiple-destination work in [2], [3], but uses network coding to achieve exact and fast topology inference.

Independently, the field of network coding [19], [20] advocates that intermediate nodes mix packets and receivers solve a system of equations. Linear network coding essentially translates network topology to the corresponding transfer matrix from the sources to the receivers (the end-points). Recently, network coding ideas have been applied to tomography problems, in order to exploit this intimate relation between topology and end-to-end observations. In [21], [22], we revisited link-loss (but not topology) tomography using active probing and network coding. In the first part of this paper, we extend our preliminary work in [1], where we showed that active probes from two sources and `XOR` at intermediate nodes are sufficient to infer the topology of a binary tree. This approach generalizes to general trees, but not to general graphs. In [23], we used a different approach for general graphs, which is closer to the work by Rabbat et al. [2], [3]: we identify 2-by-2 components and merge them together in an M-by-N topology. This journal paper combines and extends our preliminary work in [1], [23].

The following papers consider random network coding, for the purpose of information transfer, and perform *passive* topology inference on the side. In [24], passive techniques have been used to distinguish among failure patterns. In [25]–[27], subspace properties at various nodes have been used for topology inference and error localization. In [25], [28], [29], each node passively infers the upstream network topology at no cost to throughput but at high complexity. In contrast, we propose *active* probing and a simple coding scheme at intermediate nodes, to achieve low-complexity topology inference at the end nodes.[1] Furthermore, we do not require the end-points to have any a-priori knowledge of identity or operations of the intermediate nodes. In Section VII, we provide a detailed comparison and make connections between active and passive topology inference.

Finally, the predominantly employed approach to Internet topology inference today is based on `traceroute` [30]–[39]. Multiple `traceroute`'s are sent among monitoring hosts, they record router ids along paths, and this information is put together to reconstruct the graph. The `traceroute`-based approach is discussed in detail in Section VII-D.

III. Problem Statement and Model

Topology. In the first part of the paper, we consider undirected trees: there are n vertices, $n-1$ edges that can be used in both directions, and exactly one path between any two vertices. We denote by $\mathcal{L} = \{1, 2, ..., L\}$ the leaf-vertices of the tree, which correspond to end-hosts that can act as sources or receivers of probe packets.

In the second part, we consider directed acyclic graph (DAG) topologies between M sources and N receivers, which we refer to as *M-by-N topology*, following the terminology of [2], [3]. W.l.o.g., we present most of our discussion in terms of $M = 2$, *i.e.,* inferring a 2-by-N topology; an M-by-N topology can be constructed by merging smaller structures.

Similarly to [2], [3], we also assume that a predetermined routing policy maps each source-destination pair to a single route from the source to the destination. This implies the following three properties, first stated in [2].

A1 There is a unique path from each source to each receiver.

A2 Two paths from the same source to different receivers take the same route until they branch, so that all 1-by-2 components have the "inverted Y" structure; the node

[1] We would like to note that, although we present our scheme as an active scheme, in this paper, it can potentially be implemented as a passive scheme, if network coding operations are chosen to meet both network coding (*e.g.,* independence w.h.p) and tomographic (maintain the properties described in Section VII) goals at the same time.

where the paths to the two receivers split is called a *branching point*, B.

A3 Two paths from different sources to the same receiver use exactly the same set of links after they join, so that all 2-by-1 components have the "Y" structure; the node where the paths from the two sources merge is called a *joining point*, J.

These properties are consistent with the routing behavior in the Internet: the next hop taken by a packet is determined by a routing table lookup on the destination address. Each subnetwork from one source to the N receivers forms a 1-by-N tree; thus, the general graph is a "multiple-tree" network [2].

We are interested in inferring *logical* topologies, which are specified by the branching and joining points where the measured end-to-end paths meet. Intermediate nodes in a logical topology have degree at least three, and in-degree and out-degree at least one. Because this is necessary for identifiability, focusing on logical topologies is a standard assumption in topology inference problems.

Delay and loss. Link delay has a fixed part, *e.g.,* the propagation and transmission delay, and a random part, *e.g.*, the queueing delay. Path delay is the sum of delays across the links in the path. In our simulations, we consider US-wide Internet topologies, with link delays up to tens of milliseconds (ms). We assume a coarse synchronization (*i.e.,* on the order of 5-10ms) across network nodes, which is easily achievable via a handshaking scheme such as NTP. The rationale is to allow sources and intermediate nodes to operate in timeslots, of duration T and W, respectively, which are quite longer intervals than link delays. We also consider scenarios with and without packet loss.

Problem Statement. Our goal, in this paper, is to design active probing schemes, *i.e.*, the operation of sources, intermediate nodes and receivers, that will allows us to infer the logical topology from the observations at the receivers.

We restrict the space of possible operations to the simple options described in the rest of this section. In later sections, we design schemes based on these simple operations and we show that they are sufficient for topology inference. We will revisit the problem statement and make it more precise in the sections for trees and DAGs.

Operation of Sources. A pair of sources S_1 and S_2 multicast a packet each ($x_1 = [1,0]$ and $x_2 = [0,1]$, respectively) to all N receivers. More generally, sources may send symbols from a finite field $\mathbf{F}_q$. Sources send up to $countMax$ rounds of coordinated probes, which we call experiments. Experiments are spaced apart by a large interval T, to ensure that only probes in the same experiment are combined together.

Operation of Intermediate nodes. Intermediate nodes are assumed to support unicast, multicast and the simplest possible network coding operation, namely addition over a finite field $\mathbf{F}_q$. They operate in time slots of pre-determined duration or window W: a node waits for W to receive probe packets from its incoming links; if it receives more than one probe packet, then it codes them together; and it forwards (unicast or multicast) the resulting probe downstream. The choice of W affects where the probes from the two sources meet. *E.g.,* by choosing W to be larger than the maximum link delay, we can make sure that packets meet, in a hop-by-hop manner, and are coded together.

Essentially, an intermediate node can act either as a joining point (J), in which case it adds multiple incoming probe packets and forwards the result downstream; or as a branching point (B), in which case it multicasts the single incoming packet downstream. This operation will be specified more precisely in the sections for trees and DAGs.

Operation of Receivers. Each receiver i receives probes R_i, which are the source packets x_1, x_2, or a linear combination of x_1 and x_2, as the result of network coding operations at intermediate nodes. Inference of topology is based only on the observations R_i's.

Intuition. Multicast as well as network coding (which, in this paper, is limited to simple addition, thus can be thought of as reverse multicast) introduce topology-dependent correlation in the content of packets. The content of the observations at the receivers can be used to reveal the underlying topology, and in particular, the branching and joining points.

IV. Inferring Trees

Overview. We design algorithms for inferring undirected tree topologies, based only on probes sent between leaf nodes. We follow a hierarchical, top-down approach, by iteratively dividing the tree topology into smaller clusters and revealing how the groups are connected to each other.

Operation of Sources and Receivers. In each iteration (timeslot $T >> W$) a set of leaves (different across timeslots) are chosen to act as sources and the remaining leaves act as receivers. Each source sends a distinct packet. The receiver stores the first packet it receives, and discards any subsequent packets (in the same iteration).

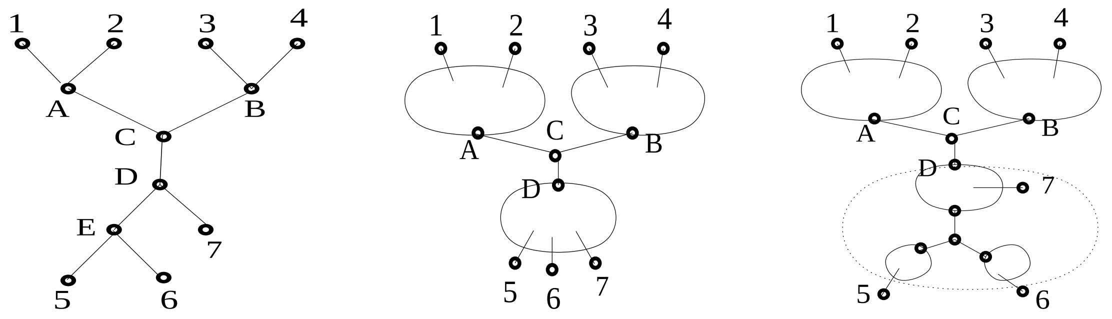
Operation of Intermediate Nodes. Every intermediate node operates in intervals of duration W. If, within W, the node receives a single probe from one of its neighbors, it multicasts it to all other neighbors. If, within W, it receives more than one packet from different neighbors, it adds them and forwards the result to all remaining neighbors. In binary trees, this linear combination is simply `XOR`. In general trees, we need operations over higher fields.

Summary of Results. In the rest of this section, we first consider binary trees, with or without packet loss. Then we extend our algorithms to m-ary trees. For trees without loss, we design deterministic algorithms that infer the topology in $O(n)$ time. For trees with loss, just one successfully received probe per network path is sufficient, without the need to collect packet loss statistics, a property that enables rapid discovery of the underlying topology.

A. Binary Trees

1) Lossless Binary Tree: Let us first consider the simplest case: an undirected binary tree without packet loss. The following example illustrates the main idea.

Example 1: Consider the tree shown in Fig. 1(a), with 7 leaves (1,2, ...7) and 5 intermediate nodes (A,B,C,D,E). Assume that nodes 1 and 7 act as sources S_1 and S_2 and



(a) Undirected binary tree we want to infer.

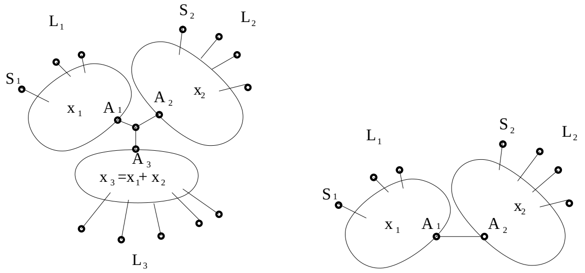
(b) Structure revealed after one iteration. Leaves 1 and 7 act as sources. Probes meet at C.

(c) Structure revealed after two iterations. Leaves 5 and 6 act as sources. Probes meet at E.

Fig. 1. Example 1: inferring the topology of an undirected binary tree with 7 leaves (1,2, ...7) and 5 intermediate nodes (A,B,C,D,E).

Algorithm 1 Topology Inference for Lossless Tree

```
Procedure ($\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, $A_1, A_2, A_3$)=SendTwoProbes($\mathcal{L}$):
  • Randomly choose two leaves in $\mathcal{L}$ to act as sources $S_1$, $S_2$ and send
    probes $x_1$, $x_2$ respectively. All other leaves $\mathcal{L} - \{S_1, S_2\}$ act as
    receivers. Intermediate nodes act as branching or joining points in trees.
  • When all receivers receive a probe, partition $\mathcal{L}$ into $\mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3$
    as follows. Set $\mathcal{L}_1$ contains $S_1$ and all receivers that observe $x_1$. $\mathcal{L}_2$
    contains $S_2$ and all receivers that observe $x_2$. $\mathcal{L}_3$ contains all receivers
    that observe $x_3 = x_1 \oplus x_2$.
  • If $\mathcal{L}_3$ is not empty, replace the original graph with the three components
    $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, connected through three edges to node $A_i$ (as in Fig. 2(a)).
    If $\mathcal{L}_3$ is empty, replace the original graph with two components $\mathcal{L}_1$ and
    $\mathcal{L}_2$, connected through a single edge (as in Fig. 2(b)) and set $\mathcal{L}_3 = \emptyset$.
  • return the components $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ and the nodes $A_1, A_2, A_3$ that
    connect each component to the network.
InferBinaryTree:
  • Call ($\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, A_1, A_2, A_3$)=SendTwoProbes($\mathcal{L}$), where $\mathcal{L}$ are all
    the leaves.
  • For each of the previously identified components $\mathcal{L}_i, i = 1, 2, 3$:
      – if $\mathcal{L}_i$ contains one or two leaves, replace the component with either
        one or two edges, connecting the leaves through node $A_i$ to the
        rest of the network.
      – if $\mathcal{L}_i$ contains three or more leaves, then call
        SendTwoProbes($A_i \cup \mathcal{L}_i$) to reveal its structure. Node
        $A_i$ that connects $\mathcal{L}_i$ to the network acts as an aggregate receiver.[3]
  • Replace vertices of degree two with a single node.
```



(a) Dividing $\mathcal{L}$ into three components.

(b) Dividing $\mathcal{L}$ into two components

Fig. 2. Edges and vertices of the graph, as revealed by a single iteration (call of SendTwoProbes) in Alg. 1. The leaves $\mathcal{L}$ are partitioned into two or three groups, based on their observations, $x_1, x_2, x_1 \oplus x_2$

send probes $x_1 = [1, 0]$ and $x_2 = [0, 1]$, respectively. The rest of the leaves act as receivers. Intermediate node A receives x_1 and forwards it to leaf 2 and to node C. Similarly, intermediate node D receives x_2 and forwards it to node E (which in turn forwards it to leaves $5, 6$) and to node C.

Probes x_1 and x_2 arrive at node C. Node C adds them, creates the packet $x_3 = x_1 \oplus x_2 = [1, 1]$, and forwards x_3 to node B, which in turn forwards it to leaves $3, 4$.[2]

At the end, leaf 2 receives x_1, leaves $5, 6$ receive x_2 and leaves $3, 4$ receive $x_3 = x_1 \oplus x_2$. Thus the leaves of the tree can be partitioned into three sets: $\mathcal{L}_1$ containing S_1 and the leaves that received x_1, *i.e.,* $\mathcal{L}_1 = \{1, 2\}$; $\mathcal{L}_2 = \{5, 6, 7\}$ containing S_2 and the leaves that received x_2; and $\mathcal{L}_3 = \{3, 4\}$ containing the leaves that received $x_1 \oplus x_2$. From this information observed at the edge of the network, we can deduce that the binary tree has the structure depicted in Fig. 1(b): three components, each seeing a different probe $(x_1, x_2, x_1 \oplus x_2)$ flowing through it, and connected through 3 links to the middle node C. This concludes the first experiment/iteration.

To infer the structure that connects leaves $\{5, 6, 7\}$ to node C, we need a second experiment. We randomly choose two of these three leaves, *e.g.,* nodes 5 and 6, to act as sources S_1 and S_2. Any probe packet leaving node D will be multicast to all the remaining leaves of the network, *i.e.,* nodes $\{1, 2, 3, 4\}$ observe the same packet. One can think of node D as a single "aggregate-receiver", which observes the common packet received at nodes $\{1, 2, 3, 4\}$. Following the same procedure as before, assuming that x_1 and x_2 meet at node E, nodes 7 and $\{1, 2, 3, 4\}$ receive packet $x_3 = x_1 \oplus x_2$. Using this additional information and the fact that the topology is a binary tree, we refine the inferred structure from Fig. 1(b) to Fig. 1(c). ■

The algorithm for inferring any binary tree is shown in Alg. 1 and generalizes the previous example. It starts by considering all the leaves of the tree $\mathcal{L}$. It calls SendTwoProbes and partitions the leaves into smaller sets/areas $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$. It proceeds by recursively calling SendTwoProbes within each set, until all edges are revealed.

Lemma 4.1: Alg. 1 terminates in at most n iterations and exactly infers the topology of an undirected binary tree.

[2] We have chosen the directionality of the edges depending on which source reaches the intermediate node first. In this example, we assume that all links have the same delay. For different delays, x_1, x_2 could meet at different nodes, but the algorithm will still work, as discussed after Lemma 4.1.

[3] Although we cannot directly observe A_i, whatever is received by A_i will be received by the leaves that are in $\mathcal{L}$ but not in $\mathcal{L}_i$; thus acting as an "aggregate" receiver on their behalf.

Proof: Consider a particular iteration (call of `SendTwoProbes`): sources S_1 and S_2 send exactly one probe packet each to all other leaves. Now consider the intermediate nodes on the path $\mathcal{P}$ between the two sources. Depending on the link delays, there are two possibilities.

The first possibility is that x_1 and x_2 meet (arrive within the same W) at one of the internal nodes on $\mathcal{P}$, *e.g.*, node A. Node A forwards their `XOR` to its third link, and the iteration reveals the neighboring edges and nodes to A as depicted in Fig. 2(a). An alternative possibility is that packets x_1 and x_2 cross each other while traversing the same link of $\mathcal{P}$ in opposite directions, *i.e.*, they do not meet at a node. Even if a leaf node receives more than one probe packets, we designed their operation so that they only keep the first one. In this case we infer the configuration in Fig. 2(b) that reveals one edge.

In summary, the algorithm iteratively divides the binary tree into smaller components until one component has two or less leaves, in which case we know its structure. In each iteration, we reveal three edges or one edge. At the end, we have revealed all $n-1$ edges. Therefore, the algorithm needs between $\frac{n-1}{3}$ and $n-1$ iterations. □

Notes. In each iteration, every link is traversed exactly once by a probe. Link delays affect where the probes meet and thus what components are revealed in each iteration. However, they do not affect the correctness of the algorithm.

2) Lossy Binary Tree: Packet loss may causes confusion when dividing the receivers into components. One solution is to send multiple probes from the same two sources during each iteration as we discuss next. However, given packet loss and delay variability, this might result in probes meeting at different nodes in the same iteration[4]. This problem is effectively created by the fact that we deal with undirected graphs, where a link may be traversed in opposite directions by probes sent in the same iteration. We can avoid this problem by fixing the directionality of the edges during each iteration. This can be achieved in a distributed manner by the first packet arriving at each intermediate node. In summary, we modify the operations of intermediate nodes as follows.

Intermediate Node Operation: Each intermediate node keeps a table of its neighbors. In each iteration, it marks these neighbors as source or sink neighbors. Once this marking is done, it does not change for the duration of the iteration. The first time during an iteration that an intermediate node receives a probe packet, the node waits for a window W to receive probes from other neighbors. After this time W passes, the node marks all neighbors from which it received packets as sources and all other neighbors as sinks. For the remaining duration of the iteration, the node accepts packets only if they originate from its source neighbors. If an intermediate node receives a packet from one of its adjacent source neighbors, it forwards it to all its sink neighbors. If it receives more than one packet from two different source neighbors, it linearly combines them, and forwards the result to all sink neighbors. The node rejects probes coming from sinks, and does not forward packets towards sources.

Algorithm 2 Topology Inference for Lossy Binary Tree

- If a receiver receives only x_1, assign it to the set $\mathcal{L}_1$.
- If a receiver receives only x_2, assign it to the set $\mathcal{L}_2$.
- If a receiver receives both x_1 and x_2, or it receives an $x_1 \oplus x_2$ packet, assign it to set $\mathcal{L}_3$.
- If a node does not receive anything, randomly assign it to one of the components.
- For aggregate receiver nodes (A_i), apply the same rule using the union of the aggregate receiver observations.

Alg. 2 presents the modifications required for Alg. 1 to be able to infer binary trees with lossy links. The only difference is that in each iteration, we send M instead of one probe packets from each source. Alg. 2 has an associated probability of error, since a leaf might not receive the “correct” probe packet[5]. Note that the number of packets M required to infer the topology within a desired error probability, is much smaller than the number of packets required by methods that collect a statistically significant number of packets and perform estimation. For our algorithm to operate correctly, it suffices that each node receives *at least one* probe packet from each of the sources it is connected to (nodes in $\mathcal{L}_1$ or $\mathcal{L}_2$ are connected to one source, S_1 or S_2 respectively, while nodes in $\mathcal{L}_3$ to two sources).

B. M-ary Trees

1) Full M-ary Trees: First let us consider full m-ary trees, *i.e.*, trees with intermediate nodes that have degree exactly $m+1$, $m \geq 3$, without packet loss. Alg. 1 can still accurately infer the topology in less than n iterations.

However, it is possible to modify the algorithm and infer the topology even faster. The idea is to keep the hierarchical clustering approach but increase the number of components revealed in each iteration, either (i) by changing the intermediate nodes so that they forward different linear combinations of the incoming probes to different outgoing links; or (ii) by increasing the number of sources in each iteration.

Modification I: (two sources per iteration, coding points send different combinations to different links). When an intermediate node receives two incoming packets from two different neighbors, it deterministically generates different linear combinations, *e.g.*, $x_1 + x_2$, $x_1 + 2x_2$,... and forwards each resulting packet to a different neighbor. Therefore, when x_1 and x_2 meet at any intermediate node, the leaves of the network will be divided into $m+1$ components, depending on which probe they receive. If the probes do not meet at a node but cross each other, the leaves of the network will be divided into two components. Once a component has m or less leaves, since we have a full m-ary tree, we know its structure. Therefore, in each iteration, we reveal $m+1$ edges or one edge, and the total number of iterations is reduced to at least $\frac{n-1}{m+1}$ and at most $n-1$. Note that the probes need to be chosen from F_{m^2}, instead of F_{2^2}.

[4] This was not a problem in the lossless case. In a given iteration, since only one probe packet is generated by each source, the packets at most meet at one intermediate node.

[5] For example, in a given iteration, an error may occur either because a node does not receive any probe packet (which can be made arbitrarily small by increasing the number of probe packets M) or, because it belongs in $\mathcal{L}_3$ but happens to receive only x_1 or only x_2 packets. This probability decreases very fast as M increases, as observed in the simulations of Section VI.

Modification II: (more than two sources per iteration, coding points send the same combination to all outgoing links). Alternatively, we can use up to m sources (as per Lemma 4.2) per iteration. The probe packets are chosen from F_{2^m}, and the sources send $x_1 = [1, 0, 0, ..., 0], x_2 = [0, 1, 0, ..., 0], ..., x_m = [0, 0, 0, ..., 1]$, respectively. When an intermediate node receives k packets from different neighbors, within W, where $2 \leq k \leq m$, it simply adds them up and forwards the result to the single remaining non-source neighbor. Depending on whether the node receives k packets or only a single packet, the leaves of the network will be divided into $m+1$ or m more components; *i.e.*, in each iteration, we reveal $m+1$ or m edges. Therefore, the algorithm requires at least $\frac{n-1}{m+1}$ and at most $\frac{n-1}{m}$ iterations.

Lemma 4.2: The maximum number of sources that can be used to uniquely infer the topology of a full m-ary tree is m.

Proof: We show that if we use $m+1$ sources to infer the topology of a full m-ary tree, it cannot be uniquely identified. Consider a binary tree with three sources sending $x_1 = [1, 0, 0]$, $x_2 = [0, 1, 0]$, and $x_3 = [0, 0, 1]$, respectively, to all other leaves in the tree. Assume that the three probe packets meet at one intermediate node; thus, we divide the tree into four components, which observe x_1, x_2, x_3, and $x_1 + x_2 + x_3 = [1, 1, 1]$, respectively. Since the degree of intermediate nodes is three, we conclude that first, two of the three sources must have joined at one intermediate node, and then, their result must have joined with the third source in another intermediate node, so that they result in $x_1+x_2+x_3$ in the last component. The first two sources can be either x_1, x_2, or x_2, x_3: we cannot uniquely infer the underlying binary tree from observing these four components. The same discussion applies to larger full m-ary trees. ■

Note. In the presence of packet loss, we can apply the same argument as in Section IV-A2, *i.e.*, we can assign directions to the links during each iteration, so that our algorithms are applicable to the lossy case as well.

2) General M-ary Trees: In general m-ary trees, *i.e.*, with degree of intermediate nodes being from three up to a maximum of $m+1$, we can apply Alg. 1 and infer the tree topology in $O(n)$ iterations. We can also apply Modification I, described in Section IV-B1. The probes should be chosen from F_{m^2} since they may meet at an intermediate node of degree $m+1$. Note that we cannot apply Modification II here: since probes may meet at an internal node of degree 3, we cannot use more than two sources, although there are nodes with degree up to $m+1$.

V. Inferring Directed Acyclic Graphs (DAGs)

A. From a Single-Tree to Multiple-Tree Topologies

So far, we considered undirected trees. Let us now consider directed trees, which are a special case of DAGs.

Example 2: Assume that we assign directions to the links of the binary tree in Fig. 1(a), all from the top to the bottom. Clearly, we can no longer send probe packets in arbitrary directions in each iteration. However, we can still infer some information about the topology. Assume that we send probes from the source nodes 1 and 2, and we observe $x_1 \oplus x_2$ at the receiver nodes 5, 6, and 7. Therefore, we identify three components $\mathcal{L}_1 = \{1\}$, $\mathcal{L}_2 = \{2\}$, and $\mathcal{L}_3 = \{5, 6, 7\}$, together with the intermediate nodes A and D, and three edges $1A$, $2A$, and AD, that connect the three components together. However, we cannot obtain more information about the internal structure of the component $\mathcal{L}_3$ or any other part of the tree network. ■

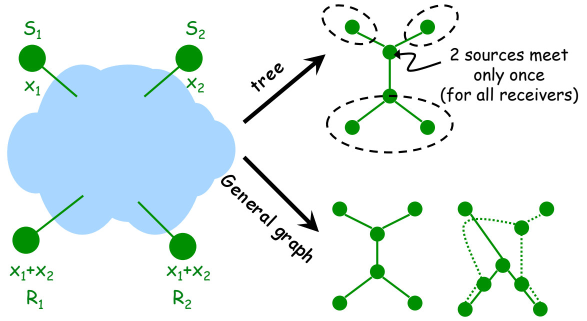


Fig. 4. Single-tree vs. multiple-tree topologies. Consider a single iteration. In a multiple-tree topology, unlike the single-tree topology, the observations at the receivers no longer uniquely identify the topology.

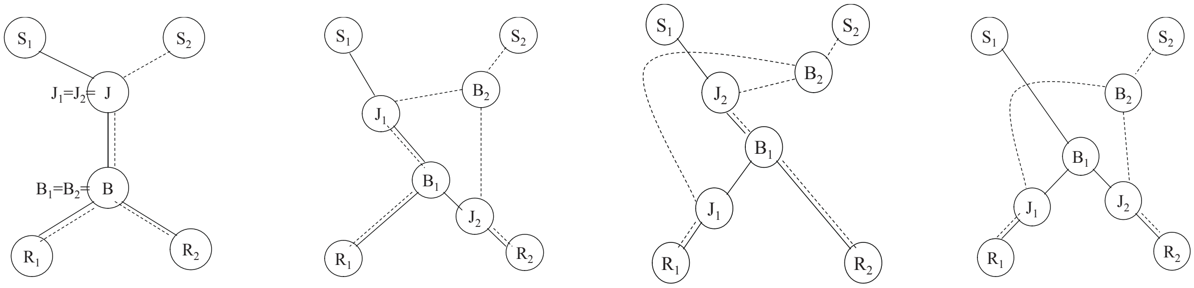
Next, consider a 2-by-2 network as defined in Section III, *i.e.*, a directed acyclic graph (DAG) with two sources, two receivers and predetermined routing. We note that directed trees are only one type among all four types of the basic 2-by-2 components of any multiple-tree network, as defined in Section III. There exist four 2-by-2 topologies, as shown in Fig. 3, which were first defined in [2], [3]. Following the same terminology as in [2], [3], we refer to Fig. 3(a), (b), (c), and (d) as type 1, 2, 3, and 4, respectively. Type 1 was called "shared" in [2], [3], since the joining points for both receivers coincide ($J_1 = J_2$) and the branching points for both sources coincide ($B_1 = B_2$). The other three types (2,3,4) are called "non-shared" since they have two distinct joining points and two distinct branching points.

In a directed tree, all 2-by-2 components are of type 1. However, in a general M-by-N topology, several different 2-by-2 types may co-exist. The algorithms described so far can identify type 1 2-by-2 topologies, and thus, trees (either completely or partially, as described above). However, they cannot distinguish between type 1 and type 4 2-by-2's.

Example 3: Consider Fig. 3 (a) and (d). Assume that in both cases, we send x_1, x_2 from S_1, S_2 to R_1, R_2 and that x_1, x_2 meet (*i.e.*, arrive within the same W) at any joining point. Therefore, in both type 1 and 4 topologies, both receivers observe $x_1 + x_2$, and we cannot distinguish between types. ■

In general, *unlike single-tree networks, the observations do not uniquely characterize the underlying topology in multiple-tree networks.* The reason is that once two sources in a tree network transmit their probe packets, they at most meet at one coding point for all the receivers, as we see in Section IV. On the other hand, in a multiple-tree network, the probe packets may meet at different coding points for different receivers, as depicted in Fig. 4. Therefore, we need a different approach.

Problem Statement. Our goal in this section is to infer a multiple-tree topology, or an "M-by-N" topology according to the terminology of Section III. Similarly to [2], we take



(a) type 1:shared (b) type 2: non-shared (c) type 3: non-shared (d) type 4: non-shared

Fig. 3. The four possible types of a 2-by-2 subnetwork component, as defined in [2]. There are two sources (S_1, S_2) multicasting packets x_1, x_2 to two receivers (R_1, R_2). (The 1-by-2 topology of S_1 is a tree composed of S_1, B_1, R_1, R_2. Similarly, The 1-by-2 tree rooted at S_2 is S_2, B_2, R_1, R_2. J_1 and J_2 are the joining points, where the paths from S_2 to R_1 and R_2, join/merge with S_1's topology.)

two steps. In the first step (Section V-B), we use several experiments and we exactly identify the type of every 2-by-2 component. In the second step (Section V-C), we merge these 2-by-2 subnetwork components to reconstruct the M-by-N network.

Operation of Sources. Pairs of sources are selected and send up to $countMax$ coordinated multicast packets to all receivers. As in the general setup, probes are spaced apart by intervals of T. In addition, we introduce a difference in the sending time of the two sources, which we call the offset u. W.l.o.g., let S_1 send first and S_2 second.

The timing parameters T, u, W are coarsely tuned so as to create observations that can distinguish among the 2-by-2 types. In particular (i) $T >> W$ ensures that only probes within the same experiment are coded together; (ii) $W >>$ link delay ensures source packets meet at the joining points despite link delays; (iii) u is selected randomly in each iteration, so that it forces probes to meet at different points, or not meet at all, in different iterations. Finally, coarse selection of T, W with rough estimates of upper bounds on link and path delays is sufficient.

Operation of Receivers. For a given 2-by-2 subnetwork, let the observations at the two receivers be $R_1 = c_{11}x_1 + c_{12}x_2$, $R_2 = c_{21}x_1 + c_{22}x_2$. Based on these observations, we design *Inference* algorithms that identify the 2-by-2 type (in Section V-B) and *Merging* algorithms that build the M-by-N from the 2-by-2's (in Section V-C).

Operation of Intermediate Nodes. In DAGs, the operation of an intermediate node, depending on whether it acts as a joining or a branching point, is summarized in Alg. 3 and Alg. 4, respectively. A joining point (J) adds and forwards packets, while a branching point (B) forwards the single received packet to all "interested" links downstream. A link is "interested" in the routing sense if it is the next hop for at least one source packet in the network coded packet.

B. Identifying 2-by-2 Components

In this section, we propose an approach for exactly identifying the type of a 2-by-2 component, using the same intuition as as in trees, *i.e.,* coding operations result in observations that can uniquely characterize the underlying 2-by-2. Our approach builds on [2] and improves over it by uniquely distinguishing among all four 2-by-2 types, while [2] was able to distinguish among shared (type 1) and non-shared (types 2,3,4) only.

Algorithm 3 Operation at Joining Point J, in DAGs. When two sources multicast to N receivers, J knows that it has two incoming links and one outgoing link. Additions are over $\mathbf{F}_q$.

```
for every time window $W$ do
    if ($J$ receives 2 packets within $W$ from its incomings) then
        as soon as the last one arrives, it adds them up, and forwards the
        resulting packet downstream
    else if ($J$ receives only one packet within $W$) then
        it forwards the packet downstream
    else if ($J$ does not receive any packet within $W$) then
        /*nothing to do*/
    end if
end for
```

Algorithm 4 Operation at Branching Point B, in DAGs. While two sources multicast to N receivers, B has one incoming packet and multiple outgoing links.

```
for each incoming packet do
    if the incoming packet is $x_1$ (or $x_2$) then
        forward it only on the outgoing links that are next hops for $S_1$ ($S_2$)
    else
        /* The incoming packet is of the form $ax_1 + bx_2$. */
        forward the packet to all outgoing links
    end if
end for
```

1) Lossless 2-by-2: First, we provide an algorithm for identifying the type of a 2-by-2 component without loss. In the first experiment, sources S_1, S_2 multicast probe packets x_1, x_2 to R_1, R_2. We begin with the assumption that S_1, S_2 act simultaneously, or in practice within the synchronization offset. A choice of large W guarantees that x_1 and x_2 meet at both joining points J_1, J_2 that add the incoming packets over $\mathbf{F}_3$. Depending on the underlying 2-by-2 type, R_1, R_2 will observe one of the following pairs:

- type 1: R_1: $x_1 + x_2$, R_2: $x_1 + x_2$
- type 2: R_1: $x_1 + x_2$, R_2: $x_1 + 2x_2$
- type 3: R_1: $x_1 + 2x_2$, R_2: $x_1 + x_2$
- type 4: R_1: $x_1 + x_2$, R_2: $x_1 + x_2$

Types 2 and 3 result in unique observations that make them distinguishable from any other type; *i.e.,* one such observation suffices to identify type 2 or 3. However, types 1 and 4 result in the same pair of observations; therefore, we need to design different experiments to get observations that can uniquely

TABLE I

Lossless Case. POSSIBLE OBSERVATIONS FOR TYPES 1 AND 4 2-BY-2 TOPOLOGIES. (OBSERVATION #1 OCCURS WHEN THE SOURCES ARE SYNCHRONIZED. OBSERVATIONS #2-4 OCCUR WHEN S_2 SENDS AFTER S_1, BY AN OFFSET $u \in [f \cdot W, W]$.)

Observation Number	Type (1)		Type (4)	
	$\mathbf{R_1}$	$\mathbf{R_2}$	$\mathbf{R_1}$	$\mathbf{R_2}$
1	$x_1 + x_2$	$x_1 + x_2$	$x_1 + x_2$	$x_1 + x_2$
2	x_1	x_1	x_1	x_1
3			$x_1 + x_2$	x_1
4			x_1	$x_1 + x_2$

Algorithm 5 Lossless Case - Inferring a 2-by-2 component. Sources S_1, S_2 multicast x_1, x_2. Receivers observe $R_1 = c_{11}x_1 + c_{12}x_2$ and $R_2 = c_{21}x_1 + c_{22}x_2$.

```
n=1; /*first experiment*/
if $c_{22} > c_{12}$ then
    Output type 2.
else if $c_{22} < c_{12}$ then
    Output type 3.
else
    /*It is $R_1 = R_2$*/
    while $n < countMax$ & $R_1 == R_2$ do
        Draw offset $u$ uniformly at random out of $[f \cdot W, W]$.
        Send probes; $S_2$ transmits $u$ time later than $S_1$.
        if $R_1 \neq R_2$ then
            Output type 4; Exit;
        end if
        n++;
    end while
    Output type 1; /* It is $n == countMax$*/
end if
```

characterize type 1 or type 4.

In the next experiment, we exploit the observation, first made in [2], that type 1 is the only 2-by-2 where the two joining points coincide ($J_1 = J_2 = J$). Therefore, the observations at the two receivers are always the same: either $x_1 + x_2$ when the two packets meet at J; or a single packet (x_1 or x_2) when the two packets do not meet at J. In contrast, type 4 has two different joining points $J_1 \neq J_2$. If we force packets to meet only at one of the joining points but not at the other one, the receivers will have different observations. These are observations #3 and #4 in Table I and they can uniquely characterize type 4.

These observations can be achieved by appropriately selecting the offset u in the sources' sending times, *i.e.*, u needs to be large enough so that after addition to the link delays, it can affect W: if D_1, D_2 are the delays on the paths from S_2 to J_1, J_2 respectively, then u must be in $[W - D_1, W - D_2]$[6].

Alg. 5 summarizes the experiments we perform in order to infer the type of a 2-by-2 network. Types 2, 3 are identified in the first observation. Type 4 is identified the first time that the two receivers see different observations. If after $countMax$ trials, we still have not seen any different observations at the two receivers, then we declare the 2-by-2 to be of type 1.

Choosing $countMax$. The larger the number of experiments, $countMax$, the smaller the error probability. Define $X = I\{R_1 = R_2\}$ to indicate whether the two receivers see the same observation. X is a Bernoulli random variable with probability $p = Pr\{R_1 = R_2\} \in [0, 1]$. The number of experiments we need to distinguish between types 1 and 4 is a geometric random variable, which stops the first time that we get different observations $R_1 \neq R_2$. If the 2-by-2 is of type 1, then $X = 1$ always, $p = 1$, and Alg. 5 cannot do a mistake. The only possible error is to mistakenly declare a type 4 topology as type 1. Assume that the two receivers had similar observations in $countMax$ trials, *i.e.*, $(X_1, X_2, ..., X_{countMax}) = (1, 1, ..., 1)$, Alg. 5 infers the 2-by-2 as type 1. Let $\alpha\%$ be the probability of correct decision after $countMax$ experiments:

$$Pr(type = 1|1, 1, ..., 1) = \frac{1}{1 + Pr(1, 1, ..., 1|type = 4)} = \alpha\% \tag{1}$$

Where we assume that the underlying topology is equally likely to be type 1 or type 4. We have that: $Pr(1, 1, ..., 1|type = 4) = [Pr(X = 1|type = 4)]^{countMax}$. Also recall that the offset u is drawn uniformly at random from $[f \cdot W, W]$ (as described in footnote 6). Therefore,

$$Pr(X = 1|type = 4) = Pr(u \notin [W - D_1, W - D_2]) = 1 - \frac{|D_1 - D_2|}{(1 - f) \cdot W}$$

$|D_1 - D_2|$ can be as small as 0, therefore, $1 - \frac{|D_1 - D_2|}{(1-f) \cdot W}$ (the probability of having similar observations in type 4) can be close to 1. For Eq. (1) to hold for $\alpha = 99\%$, we need $countMax \cong 458$. This is a pessimistic upper bound: simulations in Section VI show that $countMax$ is much smaller in practice.

2) Lossy 2-by-2: Let us now consider a 2-by-2 network where packets may be lost on some links. In this case, we can no longer guarantee meetings of x_1 and x_2 at the joining points and predictable observations at the receivers. There are two differences from the lossless case. First, because of random packet loss, each experiment might result in different outcomes, shown in Table II. Second, there are common observations across all four types, as opposed to just between types 1 and 4. We divide the observations in Table II into three groups: (i) at least one of the receivers does not receive any packet ("-") due to loss, (ii) both receivers have the same observation $R_1 = R_2$, and (iii) the two receivers have different observations $R_1 \neq R_2$.

We choose to ignore the observations of *group (i)* because they can occur in any of the four 2-by-2 types and thus do not help to distinguish among them in the deterministic way adopted in this paper. Observations of *group (ii)* can also be the result of any 2-by-2 type: unlike the lossless case, where $R_1 = R_2$ is unique to type 1 or 4 topologies, any of the four topologies may result in such observations if some packets are lost. We observe that group (ii) are the only possibility for type 1 topology, apart from the group (i) that we ignore, while all other three 2-by-2 types may result in either $R_1 = R_2$ or $R_1 \neq R_2$. Therefore, if after $countMax$ trials, we have only observations from group (ii), we declare the topology to be type 1.

In observations of *group (iii)*, it is $R_1 \neq R_2$, which means that $c_{12} \neq c_{22}$ and/or $c_{11} \neq c_{21}$. An important observation is that the difference of the coefficients between the two receivers

[6] In 2-by-2 components, this interval is close to W since D_1, D_2 are small compared to W. In more general 2-by-N networks that we consider for our simulations, there exist multiple links between the sources and joining points, link delays are on the order of tens of ms, and W is in on the order of hundreds of ms. Therefore, we safely can choose $u \in [f \cdot W, W]$ in the general case, where $0 < f < 1$ is a tunable parameter. We choose $f = \frac{1}{2}$ in our simulations, to force different observations at the two receivers.

TABLE II
Lossy Case. POSSIBLE OBSERVATIONS FOR ALL FOUR TYPES OF 2-BY-2 TOPOLOGIES. (SOURCES SEND SYNCHRONIZED AND W IS LARGE. OBSERVATION #13 FOR TYPES 2 AND 3 OCCURS ONLY WHEN S_2 SENDS WITH OFFSET $u \in [f \cdot W, W]$ AFTER S_1.) WE DIVIDE THE OBSERVATIONS INTO THREE GROUPS: (I) AT LEAST ONE RECEIVER DOES NOT RECEIVE ANY PACKET (II) $R_1 = R_2$ (III) $R_1 \neq R_2$.

Obs.	Obs.	Type 1		Obs.	Type 2		Obs.	Type 3		Obs.	Type 4	
#	Group	$\mathbf{R_1}$	$\mathbf{R_2}$	Group	$\mathbf{R_1}$	$\mathbf{R_2}$	Group	$\mathbf{R_1}$	$\mathbf{R_2}$	Group	$\mathbf{R_1}$	$\mathbf{R_2}$
1	(i)	-	-	(i)	-	-	(i)	-	-	(i)	-	-
2		-	$x_1 + x_2$		-	$x_1 + 2x_2$		$x_1 + 2x_2$	-		-	$x_1 + x_2$
3		-	x_1		-	$x_1 + x_2$		$x_1 + x_2$	-		-	x_1
4		-	x_2		-	x_1		x_1	-		-	x_2
5		$x_1 + x_2$	-		-	x_2		x_2	-		$x_1 + x_2$	-
6		x_1	-		$x_1 + x_2$	-		-	$x_1 + x_2$		x_1	-
7		x_2	-		x_1	-		-	x_1		x_2	-
8	(ii)	$x_1 + x_2$	$x_1 + x_2$		x_2	-		-	x_2	(ii)	$x_1 + x_2$	$x_1 + x_2$
9		x_1	x_1	(ii)	$x_1 + x_2$	$x_1 + x_2$	(ii)	$x_1 + x_2$	$x_1 + x_2$		x_1	x_1
10		x_2	x_2		x_1	x_1		x_1	x_1		x_2	x_2
11					x_2	x_2		x_2	x_2	**(iii)**	$\mathbf{x_1}$	$\mathbf{x_1 + x_2}$
12				**(iii)**	$\mathbf{x_1 + x_2}$	$\mathbf{x_1 + 2x_2}$	**(iii)**	$\mathbf{x_1 + 2x_2}$	$\mathbf{x_1 + x_2}$		$\mathbf{x_1 + x_2}$	$\mathbf{x_1}$
13					$\mathbf{x_1}$	$\mathbf{x_1 + x_2}$		$\mathbf{x_1 + x_2}$	$\mathbf{x_1}$		$\mathbf{x_1}$	$\mathbf{x_2}$
14					$\mathbf{x_1}$	$\mathbf{x_2}$		$\mathbf{x_2}$	$\mathbf{x_1}$		$\mathbf{x_2}$	$\mathbf{x_1}$
15					$\mathbf{x_1 + x_2}$	$\mathbf{x_2}$		$\mathbf{x_2}$	$\mathbf{x_1 + x_2}$		$\mathbf{x_1 + x_2}$	$\mathbf{x_2}$
16											$\mathbf{x_2}$	$\mathbf{x_1 + x_2}$

contains topology-related information. W.l.o.g, we focus on the coefficient of x_2 and look at the difference $c_{12} - c_{22}$. Table II shows that $c_{12} - c_{22} < 0$ can only occur in type 2 or type 4 topologies; while $c_{12} - c_{22} > 0$ can only occur in a type 3 or 4 topology. Note that the coefficient is larger on one side (*e.g.*, $c_{12} > c_{22}$) when the probe (x_2) goes through two joining points on its way to one receiver (in this case, R_1) and through one joining point on its way to the other receiver (R_2). By performing several independent experiments and collecting several observations of group (iii), we can distinguish among the candidate topologies. If after $countMax$ experiments, there are only observations of group (ii) or (iii) with $c_{12} - c_{22} \leq 0$, we declare the topology as type 2. If there are only observations of group (ii) or (iii) with $c_{12} - c_{22} \geq 0$, we declare it as type 3. If there are observations of group (ii) or (iii) with both $c_{12} - c_{22} < 0$ and $c_{12} - c_{22} > 0$, we declare it as type 4.

In our experiments, we try to create those observations that reveal the topology. These can occur either naturally, as the result of packet loss, or artificially, by us introducing an offset u in S_2's sending time with respect to S_1. To help these observations occur, especially for small loss rate, and similarly to the lossless case, we use a random offset $u \in [f \cdot W, W]$. To make these experiments independent, we space apart successive sets of probes by roughly selecting $T \geq 3W$, which is sufficient since there are at most two joining points on any (S_i, R_j) path in a 2-by-2.

Alg. 6 summarizes the 2-by-2 inference for lossy networks. The algorithm is simple and follows a deterministic approach: one observation, or a set of observations, is sufficient to uniquely distinguish among types. For example, at least one observation of group (iii) rules out the type (1) topology; a pair of group (iii) observations with both $c_{12} - c_{22} > 0$ and $c_{12} - c_{22} < 0$ indicates type 4; etc. As a result, we require less experiments compared to thousands of arrival order measurements required by [2], [3] for statistical significance. In addition and more importantly, we identify the exact 2-by-2 type while [2] only distinguishes between shared and non-shared.

Algorithm 6 Lossy Case - Inferring a 2-by-2 component. Sources S_1, S_2 multicast x_1, x_2. Receivers observe $R_1 = c_{11}x_1 + c_{12}x_2$ and $R_2 = c_{21}x_1 + c_{22}x_2$. The variable *type* stores our estimate of the type of the 2-by-2 compoenent and it gets updated during the experiments.

```
$n = 1$; /*first experiment*/
type=0; /*initialization*/
while $n \leq countMax$ do
    if $R_1 \neq [0,0]$ & $R_2 \neq [0,0]$ then
        if $c_{22} > c_{12}$ then
            if type $\neq 3$ then
                type=2;
            else
                type=4; Break;
            end if
        else if $c_{22} < c_{12}$ then
            if type $\neq 2$ then
                type=3;
            else
                type=4; Break;
            end if
        else if $type == 0$ & $R_1 == R_2$ then
            type=1;
        end if
    end if
    n++;
    Draw offset $u$ uniformly at random out of $[f \cdot W, W]$.
    Send probes; $S_2$ transmits $u$ time later than $S_1$.
end while
Output type.
```

3) *Inferring all 2-by-2's in a 2-by-N Network:* Algorithms 5 and 6 can be directly applied to a 2-by-N network, *i.e.*, a network where two sources multicast to N receivers. A difference is that intermediate nodes need to perform addition over a larger finite field (of order larger than the maximum number of joining points on a path). Algorithm 5 and Algorithm 6 can be performed on any pair of receivers among all $\binom{N}{2}$ possible pairs. The same set of 2-by-N probes can be used to infer, in parallel and independently, the type of all 2-by-2 topologies.

This reduces the number of probes, as we re-use them, instead of sending $\binom{N}{2}$ different sets of probes. The 2-by-N structure is important for the merging algorithm in Section V-C.

4) 2-by-2's vs. other Subnetwork Components: We now discuss why we choose to decompose an M-by-N network into 2-by-2 subnetwork components, as opposed to any other subnetwork structures, *e.g.,* $m-by-n$'s, where $1 \leq m \leq M, 1 \leq n \leq N$.

- 1-by-1: This is the smallest component and corresponds to measuring a single end-to-end path. However, it does not reveal neither joining nor branching points.
- 1-by-2 and 2-by-1: These correspond to a 2-leaf multicast or a reverse-multicast tree, respectively. The 2-by-1 consists of 2 sources, one coding point, 1 receiver. The 2-by-1 cannot identify the branching points while the 1-by-2 cannot identify the joining points. Similar comments apply to M-by-1 and 1-by-N.
- 2-by-2: This is the smallest structure that gives information about the relative locations of joining and branching points.
- m-by-n, with $2 < m < M, 2 < n < N$: If we consider larger structures, there is an exponentially larger number of possible types, which requires more complicated inference algorithms. *E.g.,* there exist 19 possible types for a 2-by-3.
- M-by-N: In the extreme case, we need to enumerate all possible M-by-N topologies, as in [25].

The larger the subnetwork component we use as a building block, the less components we need to infer and the simpler the merging algorithm. However, as the size of the basic component grows, the number of possible types increases exponentially and the inference step becomes increasingly complex. In this paper, we choose to decompose an M-by-N into 2-by-2 components, inspired by the approach in [2]. We note that 2-by-2 is the minimum size building block required to infer both joining and branching points and strikes a good tradeoff of inference vs. merging complexity.

C. Merging Algorithm

Assuming knowledge of all 2-by-2 subnetwork components, from Section V-B, we now merge them together to reconstruct the M-by-N network. We study merging in two different scenarios: (i) when a 1-by-N tree topology is known, which is the same problem studied in [2]; and (ii) without knowledge of any 1-by-N, which is new to our work. Exploiting the accurately identified 2-by-2's, we can solve (i) exactly, which was previously only approximately solved; and also solve (ii), which was previously not known how to address.

More precisely, our merging algorithm can identify every joining point, in the sense that it can localize it between two branching points. However, note that when there are several joining points in a row without any branching points in between, it is not possible to identify the relative locations of these joining points with respect to each other. In fact, this is the case in a tree topology.

1) Merging a 1-by-N and 2-by-2's into a 2-by-N: In this section, we assume that the 1-by-N from S_1 to N receivers is known, using any of the classic methods for single-tree topology inference, *e.g.,* see [4], or our algorithms in Section IV for tree networks. This 1-by-N is a tree rooted at S_1 and contains only branching points. We also assume that the 2-by-2's between S_1, a new source S_2, and any pair of receivers are known, using the algorithms of Section V-B. Our goal is to locate the joining points where paths from S_2 to the same N receivers join S_1's topology. We use the assumptions of Section III for routing.

Algorithm 7 Merging Algorithm: Given the two sources S_1 and S_2, a set of receivers $R_1, R_2, ..., R_N$, the 1-by-N S_1 tree topology, and the 2-by-2 results from Algorithm 6 for any pair of receivers R_i, R_j, this algorithm identifies a single link for the location of every J_i (the joining point for R_i), on S_1 topology.

```
for each receiver $R_i$ do
    if $\exists k < i$ such that the $S_1, S_2, R_k, R_i$ 2-by-2 is shared then
        $J_i = J_k$;
    else
        Let $B$ be the closest branching point to $R_i$
        while $J_i$ is not localized to a single link do
            Let $R_j$ be any child of $B$ ($j \neq i$)
            Based on the type of the 2-by-2 component $S_1, S_2, R_i, R_j$,
            locate $J_i$ above/below $B$
            if ($J_i$ is below $B$) || (($J_i$ is above $B$) && ($\nexists$ other branching
            point above $B$ on $S_1$'s 1-by-N)) then
                $J_i$ is localized to a single link.
                Output this link; Break;
            else
                $B =$ the next upstream branching point
            end if
        end while
    end if
end for
```

This problem was posed in [2], [40] and solved there in an approximate way. Bounds on the joining points locations in the S_1 topology were provided within a sequence of consecutive logical links. This was a result of the fact that 2-by-2's are only identified as shared or non-shared types in [2], [3].

In contrast, we design Algorithm 7, which localizes each joining point for each receiver to a single logical link, between two branching points in the S_1 topology. Our algorithm is simpler, faster, and more accurate: it can identify *all* joining points for any topology and with lower complexity, thanks to our complete knowledge of the 2-by-2 types.

Example 4: Fig. 6(a) depicts a 2-by-9 topology constructed based on the Abilene network [41]. Consider R_1: it forms a type 1 2-by-2 with R_2. Thus J_1 must lie above $B_{1,2}$, so that there exists a unique path from each source to R_1. But then $B_{1,3}$ is on the way. R_1,R_3 form a 2-by-2 of type 4, thus J_1 must be below $B_{1,3}$. Now J_1 is localized to one link and the algorithm ends here for R_1. Other receivers are considered similarly. Note that a joining point can be placed on any link from the receiver to S_1, thus, the number of steps required to localize a joining point is at most the height of the S_1 tree. Also, when there is a group of receivers within which all pairs are of type 1, the algorithm is run once and assigns the same joining point to all of them. For this example, the algorithm in [2] cannot completely resolve all joining points and provides bounds within a sequence of several logical links instead. ■

2) Merging 2-by-2's into a 2-by-N: In this section, we infer a 2-by-N without prior knowledge of any 1-by-N. Inference under this relaxed assumption is enabled by our exact knowledge of 2-by-2's and was not possible before [2], [40]. We first send probes over the 2-by-N and then we merge all $\binom{N}{2}$ 2-by-2 components, as described next.

Example 5: We first consider all shared (type 1) 2-by-2 components and assign them the minimum number of branching and joining points required. For example in Fig. 6(a), $B_{1,2}, B_{3,4}$ and $J_1 = J_2, J_3 = J_4 = J_5 = J_6 = J_7 = J_8 = J_9$ are identified in this step. Second, we consider all non-shared 2-by-2 topologies (of type 2, 3, or 4). We use the information about the locations of the branching and joining points in each type to: (1) add the minimum number of branching points required to the ones already identified from the shared pairs; and (2) assign joining points to those receivers that have not been already assigned one. In the example of Fig. 6(a), an additional branching point $B_{1,3}$ is required, which is connected to both joining points $J_1 = J_2$ and $J_3 = J_4 = J_5 = J_6 = J_7 = J_8 = J_9$, to satisfy the 2-by-2's of type 4 between the two shared groups. No additional joining point is required in this example. ■

This approach identifies the locations of all joining points, between the S_1 and S_2 1-by-N topologies, but does not identify all the branching points in the S_1 tree topology. Only the "minimum" S_1 topology is identified, *i.e.,* the tree made by the "necessary" branching points. We define as "necessary" branching points the ones located below a joining point of S_1 and S_2 in the 2-by-N. An "unnecessary" branching point is the child of another branching point with no joining point in between. This approach does not identify $B_{4,5}, B_{6,7}, B_{6,9}$, and directly connects their children ($R_4, R_5, R_6, R_7, R_8, R_9$) to the upstream branching point ($B_{3,4}$).

Note that the worst case input for this approach is a tree network. Since all 2-by-2's are of type 1, and the algorithm cannot reconstruct branching points in a row, it can only identify the top-most branching point of the entire tree structure.

3) From 2-by-N to M-by-N: We can directly extend the 2-by-N inference techniques to the M-by-N case [40]. We start from a 2-by-N topology, and add one source at a time, to connect the 1-by-N's of the remaining $M-2$ sources. Assume that we have constructed a k-by-N topology, $2 \leq k < M$. To add the $(k+1)^{th}$ source, we perform k experiments, where at each experiment one different of the k sources and the $(k+1)^{th}$ source send x_1 and x_2. We then "glue" these topologies together by following the topological rules of Section V-C1.

4) Complexity of Merging:

Lemma 5.1: Identifying at least $N-1$ 2-by-2 components is necessary for Alg. 7 to be able to identify all joining points.

Proof: The main idea is provided in the following example, which is chosen to demonstrate the worst case. Alg. 7 requires the maximum number of 2-by-2's to localize all the joining points in this example topology.

Consider Fig. 5; Algorithm 7 starts from R_1, and considers the 2-by-2 type of R_1, R_2 first. Assume that R_1, R_2 form a 2-by-2 of type 1, and thus, their joining point lies above $B_{1,2}$. Therefore, we need to continue to the upper branching points. In the next step, we check the 2-by-2 type of R_1 with one child of $B_{1,i}$, *e.g.,* R_i. Assume that the joining point for R_1 still lies above $B_{1,i}$; thus, we need to continue until the highest branching point and consider R_1 with one child of $B_{1,N}$, *i.e.,* R_N. The joining point for R_1 (also R_2, R_N) is finally localized above $B_{1,N}$. We now need to find the joining point location for any other receiver $R_i, R_{i+1}, ..., R_k$. Each R_i needs to be considered with one child of any of its upper branching points until its joining point is identified, which can be up to $B_{1,i}$ at most; because otherwise it would be shared with R_1. Assume that in the worst case, R_i needs to be considered with all of them, *i.e.,* $R_iR_{i+1}, ..., R_iR_k$, and its joining point lies directly below $B_{1,i}$. For any of the remaining receivers, we need to continue in the same way; their joining points will lie below their branching point with R_i, because otherwise, they would be the same as J_i.

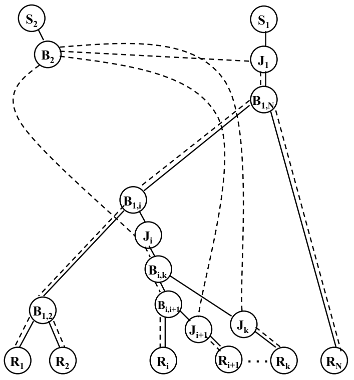


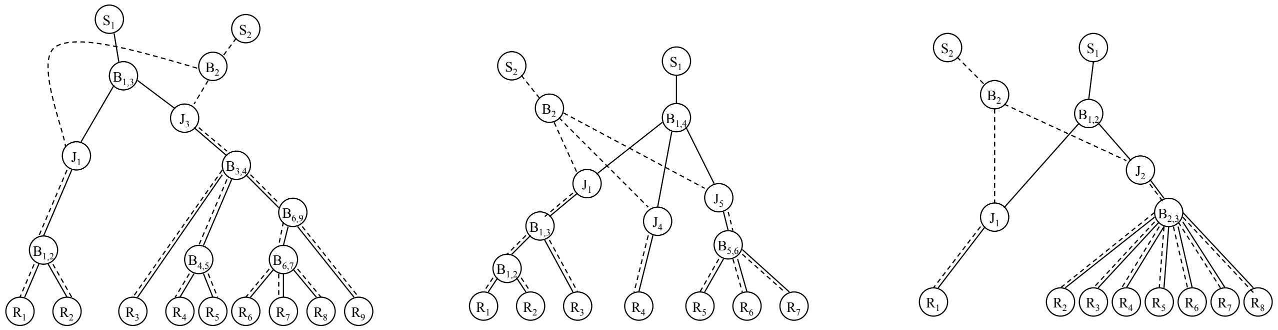
Fig. 5. Counting the 2-by-2's required for merging algorithm 7 to uniquely localize all joining points.

The following matrix shows the list of all 2-by-2's that are required to identify all joining points. It is an $N \times N$ upper-triangular matrix, where each row or column is corresponding to one of the N receivers. The non-zero elements may represent any of the four 2-by-2 types. However, we have only indicated the ones we require (with *yes*) and the ones we do not require (with *no*).

$$\begin{pmatrix} 0 & yes & no & \dots & no & yes & no & \dots & \dots & \dots & no & yes \\ 0 & 0 & no & \dots & no & no & \dots & \dots & \dots & \dots & \dots & no \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \\ 0 & 0 & \dots & 0 & 0 & 0 & yes & \dots & yes & no & \dots & no \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Intuitively, the list of the 2-by-2's required to identify the joining point for each receiver is separate from any other receiver. Therefore, the total number of required 2-by-2's is no more than the length of the longest row in the matrix above (*i.e.,* the first row), which is $N-1$. ■

Note on Lemma 5.1: If the 2-by-2's are properly selected, $N-1$ is sufficient. Unfortunately, we can not know in advance (*i.e.,* without knowledge of the 2-by-N topology) which 2-by-2's to choose out of all $\binom{N}{2}$ possible 2-by-2's, so as to uniquely localize the joining points between two branching points. Nevertheless, from the given S_1 1-by-N, we can give an upper bound on the number of 2-by-2 types required. Since



(a) The Abilene topology [41].

(b) An Erdos-Renyi random graph.

(c) A preferential attachment graph.

Fig. 6. Three different topologies used to test our inference algorithms in simulation.

every receiver needs to be checked with other receivers that are children of its upper branching points, up to the location of its joining point, we need to check for $O(N \log N)$ 2-by-2's. This is less than identifying all $\binom{N}{2}$ 2-by-2's. Note that we still need to multicast x_1, x_2 to all receivers and monitor all observations; but we can use only the observations of the selected 2-by-2's for inference and ignore the rest.

Lemma 5.2: The complexity of Algorithm 7 is O(N).

Proof: The algorithm starts from a receiver and proceeds to its upper branching points until its joining point is localized below one of the branching points or above the highest one. At each step, the algorithm uses information from one 2-by-2 to improve its estimate of the joining point location. Therefore, the total number of steps performed by the algorithm equals the number of required 2-by-2's given in the matrix above. There are also some repeated checks of the same 2-by-2's for those receivers that are shared with previously considered receivers. However, the complexity of the algorithm is still in the order of O(N). ■

Finally, one can see that the second merging algorithm needs to know all $\binom{N}{2}$ 2-by-2's, thus also takes $\binom{N}{2}$ steps.

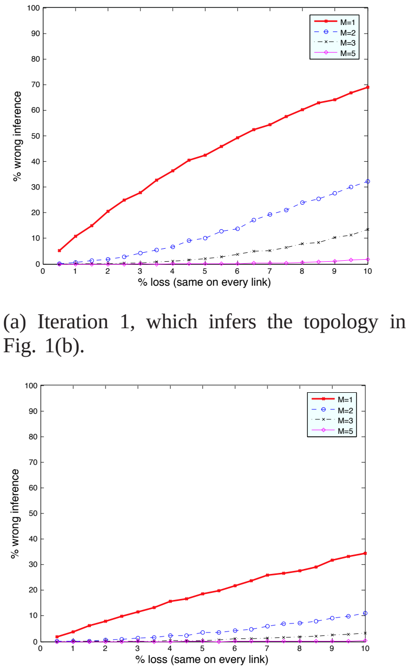
VI. Simulation Results

We now simulate our Inference[7] algorithms in some representative topologies that exemplify different characteristics.

A. Trees

1) Simulation Setup: Consider the binary tree example of Fig. 1(a). Assume that all links have the same loss probability $p \in [0, 10\%]$. We simulate Alg. 2 and we send up to $M = 10$ probes per iteration. We conservatively consider an error to be *any* divergence from the true topology. The results are averaged over $10,000$ realizations of the loss process.

2) Simulation Results: Fig. 7 shows the percentage of inference errors in each of the first two iterations (shown in Fig. 1(b) and Fig. 1(c)) as a function of p and M. As expected, the probability of error is increasing with p, since packet losses may lead to the misclassification of a leaf to the incorrect



(a) Iteration 1, which infers the topology in Fig. 1(b).

(b) Iteration 2, which infers the topology in Fig. 1(c).

Fig. 7. Probability of incorrect inference for the binary tree of Fig. 1(a), as a function of the loss probability p (same for all links) and of the number of probes M per iteration.

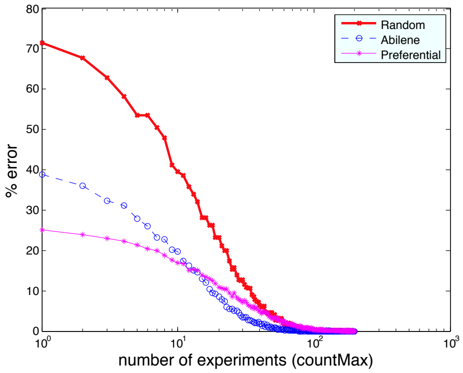
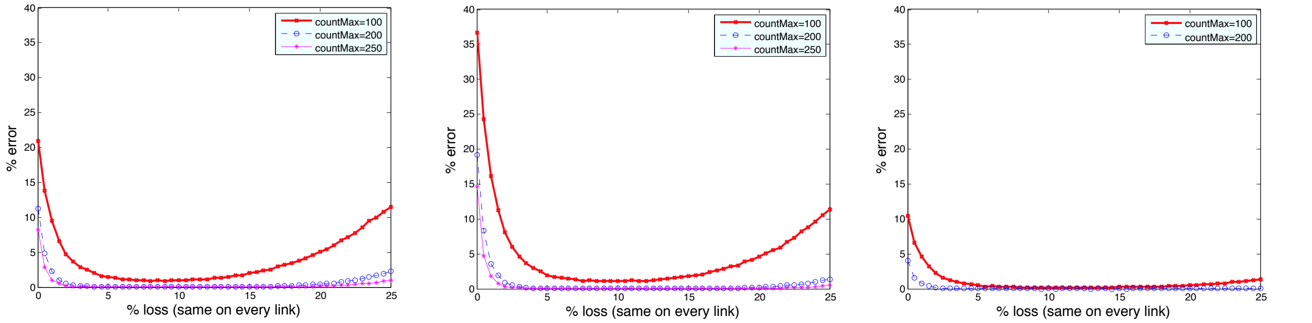


Fig. 8. Lossless case. The error probability vs. the number of experiments for the three topologies in Fig. 6, 1000 realizations.

[7] We note that, in both our approach and in past work [2], [3], the error in identifying the 2-by-2's, in the first step, may propagate to the Merging algorithm, in the next step. However, there is no additional error introduced by the Merging algorithm itself, and thus no need to simulate it.



(a) Simulation results for the Abilene topology in Fig. 6(a).

(b) Simulation results for the random topology in Fig. 6(b).

(c) Simulation results for the preferential topology in Fig. 6(c).

Fig. 9. Lossy case. The error probability vs. the loss rate for different values of $countMax$ for the three topologies in Fig. 6, 10,000 realizations.

component. For a fixed number of probes per iteration and fixed loss rate p, the error probability decreases with the iterations. Also the probability of error decreases rapidly with the number of probes M per iteration: it decreases significantly even with $M = 2, 3$, even for large p, and becomes practically zero for $M \geq 5$.[8]

B. *Multiple-Tree Topologies (DAGs)*

We simulate our algorithms in example multiple-tree networks. In summary, we show that (i) our approach significantly improves over [2], [3], [40], in terms of the number of experiments required to identify the type of all 2-by-2's as well as of the associated probability of error; (ii) the probability of error in identifying the 2-by-2's depends on the underlying topology. In particular, it is smaller for preferential attachment graphs as compared to ER random graphs.

1) Simulation Setup: To demonstrate (i), we consider Fig. 6(a), which shows the Abilene topology [41], with two sources located at the Chicago and Indiana nodes, and nine receivers, each located at one of the other core network nodes. This is the same topology considered in [2]. To investigate (ii), we consider Fig. 6(b) and Fig. 6(c). Fig. 6(b) shows a random topology with 2 sources and 7 receivers generated by LEDA [42]. Fig. 6(c) shows a preferential attachment topology generated by Brite [43]. We pick 2 sources and 8 receivers and we select the route for every source-destination pair, according to our assumptions in Section III.

We run Alg. 5 and Alg. 6 in the absence and presence of packet loss, respectively, and we compute the error. In the lossless cases, we identify the 2-by-2 types and report the error as a function of the number of experiments $countMax$. The only possible error is to falsely declare a type 4 as type 1. In the lossy case, we also report the error assuming that there is packet loss in the network (with prob. p independently on every link), and after applying Algorithm 6 to each topology. An error in this case can result either from declaring type 2 or 3 or 4 as type 1; or from declaring type 4 as type 2 or 3. We consider values of $p \in [0, 25\%]$ and $countMax = 100, 200, 250$.

We assume that individual link delays have a fixed part of 5-10ms (propagation delay), and a variable part, which is exponential with a maximum of 10ms (queueing delay). We choose a large time window $W = 100ms$. The offset u is drawn uniformly at random out from $[50, 100]ms$, *i.e.*, $f = \frac{1}{2}$.

2) Simulation Results: Fig. 8 reports the results for the lossless case, and Fig. 9 for the lossy case, for all three topologies of Fig. 6.

Let us first discuss the *Abilene topology*, shown in Fig. 6(a). In the lossless case, the Abilene curve in Fig. 8 shows that the error probability decreases very rapidly with $countmax$ and reaches 0 at $countMax \simeq 150$. In the lossy case, as shown in Fig. 9(a), the error probability also decreases rapidly with $countMax$: it becomes negligible with 200 - 250 experiments. This is a significant improvement over [2] for the same example topology: they used 1000 measurements to distinguish only between type 1 and the other three types, for very small loss rates of up to 1.5%, and they achieved error probability 10%. In contrast, with an order of magnitude less probes, we distinguish among all four types, and we have a very small error probability for larger loss rates (up to 25%). Note that the error probability is not monotonic with p: for small loss rates, Algorithm 6 results in more erroneous cases while Algorithm 5 could give better results. The effect of loss is to increase the number of observations of all three groups. However, for moderate loss rates, we get enough observations of group (iii), thus a small error probability. For larger loss rates, the increase in the observations of group (i), which we ignore, increases the error probability again, especially for small $countMax$.

Let us now consider *random graphs*, in particular Erdos-Renyi (ER) vs. Preferential Attachment, depicted in Fig. 6(b) and Fig. 6(c), respectively. In the lossless case, we can see in Fig. 8 that the error probability decreases very rapidly with $countMax$ and reaches 0 at $countMax \simeq 150$. We also observe that, for the same number of experiments, the error is generally smaller in the topology generated using the preferential attachment rather than the ER model. This is true both in the lossless (Fig. 8) and in the lossy (Fig. 9(b) and

[8] This second observation is due to the fact that *one* correctly received packet is sufficient for the correct operation of Alg. 2. *E.g.*, if a node receives a mixture of x_1 and x_2 probes, it will be correctly assigned to component $\mathcal{L}_3$ even if some probes are lost. In contrast, methods that require each receiver to receive enough probe packets to infer the probability of loss rate associated with the network links with a certain accuracy, require a larger number of probes for statistical significance.

Fig. 9(c)) case. [9]

VII. Our Work in Perspective

In this section, we revisit the related work, briefly outlined in Section II, but now focusing on the most closely related parts. We provide an in-depth comparison of the trade-offs involved and we draw connections between different approaches. We outline possible extensions that can unify active and passive tomography with network coding and directions for future work.

A. Comparison to Traditional Tomography w/o Network Coding

Within the large literature on network tomography, the most closely related work is the Multiple Source Network Tomography in [2], [3], [40], which formally defines M-by-N tomography problem. Our work on DAGs builds on [2], [40]: we follow their graceful approach for decomposing the M-by-N into a number of 2-by-2 components, inferring the type of each 2-by-2 and then merging them up together to reconstruct the M-by-N. Using simple network coding operations at intermediate nodes, provides a graceful way to reveal coding points, which has been typically a challenge in traditional tomography. Our work improves upon [2], [40], in that: (i) it can *exactly* identify the type of a 2-by-2, as opposed to just distinguish between shared and non-shared type; and (ii) the merging algorithms can precisely locate the locations of joining points with respect to the branching points, as opposed to provide bounds.

Simulation results in Section VI on the same topology used in [2], showed that our approach is more accurate, with less experiments. In essence, our approach is deterministic (one observation suffices to distinguish among types) as opposed to probabilistic (which needs to collect a large number of probes for statistical significance). This benefit comes at the cost of having intermediate nodes do some operations. However, these operations are so simple (just additions), that can be simply thought of as inverse multicast. This cost can be removed, if our approach is implemented as passive on top of random network coding, as outlined in Section VII-C.

B. Comparison to Passive Tomography with Network Coding

Recently, a passive approach for topology inference on top of random network coding has been proposed in [25], [28], [29]. The probes are sent only once, and intermediate nodes pick coding coefficients β uniformly at random out of a large field $\mathbf{F}_q$. The key idea is that, under assumptions of strong connectivity and large enough finite field, $\mathbf{F}_q$, the transfer matrix M, from the sender to the receiver, is distinct for different networks, w.h.p. Then, using the observations Y at the receivers and the source messages X, exhaustive enumeration of all possible topologies is used to find an M that matches $Y = MX$. (Note that M is directly observed at the receiver because the source adds unit vectors at the beginning of its message.)

Our approach is different in that it is active and uses several probes but simple coding operations over a small field. Furthermore, we do not require the end-points to have any a-priori knowledge of identity or operations of the intermediate nodes.

Example 6: To better illustrate the differences, let us consider a 2-by-2, and let us try to infer its type using the two approaches. The transfer matrices corresponding to the four types of a 2-by-2, shown in Fig. 3, are provided in Fig. 10. In contrast, we send probe packets in multiple rounds. In each experiment, β's are either 0 or 1 (since we do additions only), and we exclude some of the possible topologies in each experiment, until we are left with only one unique topology, in at most $countMax$ experiments. ■

Our $countMax$ experiments can be thought of as collecting observations $Y_1 = M_1X, Y_2 = M_2X, ..., Y_{countMax} = M_{countMax}X$, where $M_1, M_2, ..., M_{countMax}$ are different representations of that unique M. Note that, although M is unique in terms of β's for each topology, it can be shown to be non-unique when these β's are replaced by $0/1$ values. For example, the transfer matrices for types 1 and 4 seem to be the same when all β's are equal to 1, but only type 4 can potentially result in $M = [1, 1; 0, 1]$. We send probes in multiple experiments to create those representations of M that help us uniquely identify the underlying topology.

In terms of finite field size, [25], [29] needs a much larger finite field than us, in order to get distinct transfer matrices for different topologies.[10] In terms of bandwidth, our approach uses smaller packets in a single experiment, since our operations are performed over a smaller field and a few experiments are required. Furthermore, in [25], [29] the coefficients, sent anyway along with the packets through the network, are used to reveal the topology from the transfer matrix at the receiver, and can be thought of as the equivalent of probes. The distinction between active and passive approach becomes even less pronounced, if we consider that [25], [29] requires the receiver to have a-priori knowledge of the size of the network, and of the code-book used at each node (referred to as "common randomness"), which depends on the node ID [28]. In contrast, we do not require such knowledge and we infer the topology with smaller complexity.

Note that in the general 2-by-N case, similarly to the 2-by-2 case, we have a 2-by-N transfer matrix M, with each column corresponding to one of the receivers. We continue sending probes until we get a unique 2-by-2 topology per every pair

[9] The reason is that in the preferential attachment topologies, we have a few nodes with very high degree and many nodes with a low degree. As a result, we have a large number of receivers with a shared joining point and some other receivers with distinct joining points. In contrast, in ER graphs, we have several roughly equally-sized groups of shared receivers, where each group forms a non-shared type with another group. Therefore, we have more topologies of type 1 in preferential attachment, which results in smaller error.

[10] In [25], it was shown that if the local coding variables are i.i.d uniform r.v. over $\mathbf{F}_q$, then the probability that all different unicast networks with at most $|V|$ nodes and at most $|E|$ edges will have distinct transfer matrices $\geq 1 - |V|^{4|E|}(1-(1-\frac{1}{q})^{|V|})$. This indicates that (i) the probability of success goes to 1 iff $q \to \infty$, and (ii) q needs to increase rapidly as the size of the network grows. In contrast, as we see in Section V-B, our approach requires only a small field $\mathbf{F}_3$ to distinguish among different 2-by-2 topologies. We can calculate that if β's are chosen uniformly at random from $\mathbf{F}_3$, then $Pr(M_4 = M_1) \cong 0.04$, which is not negligible for these very small networks.

$$M_1 = \begin{pmatrix} \beta_{S_1J,JB}\cdot\beta_{JB,BR_1} & \beta_{S_1J,JB}\cdot\beta_{JB,BR_2} \\ \beta_{S_2J,JB}\cdot\beta_{JB,BR_1} & \beta_{S_2J,JB}\cdot\beta_{JB,BR_2} \end{pmatrix}$$

$$M_2 = \begin{pmatrix} \beta_{S_1J_1,J_1B_1}\cdot\beta_{J_1B_1,B_1R_1} & \beta_{S_1J_1,J_1B_1}\cdot\beta_{J_1B_1,B_1J_2}\cdot\beta_{B_1J_2,J_2R_2} \\ \beta_{S_2B_2,B_2J_1}\cdot\beta_{B_2J_1,J_1B_1}\cdot\beta_{J_1B_1,B_1R_1} & \beta_{S_2B_2,B_2J_2}\cdot\beta_{B_2J_2,J_2R_2} + \beta_{S_2B_2,B_2J_1}\cdot\beta_{B_2J_1,J_1B_1}\cdot\beta_{J_1B_1,B_1J_2}\cdot\beta_{B_1J_2,J_2R_2} \end{pmatrix}$$

$$M_3 = \begin{pmatrix} \beta_{S_1J_2,J_2B_1}\cdot\beta_{J_2B_1,B_1J_1}\cdot\beta_{B_1J_1,J_1R_1} & \beta_{S_1J_2,J_2B_1}\cdot\beta_{J_2B_1,B_1R_2} \\ \beta_{S_2B_2,B_2J_1}\cdot\beta_{B_2J_1,J_1R_1} + \beta_{S_2B_2,B_2J_2}\cdot\beta_{B_2J_2,J_2B_1}\cdot\beta_{J_2B_1,B_1J_1}\cdot\beta_{B_1J_1,J_1R_1} & \beta_{S_2B_2,B_2J_2}\cdot\beta_{B_2J_2,J_2B_1}\cdot\beta_{J_2B_1,B_1R_2} \end{pmatrix}$$

$$M_4 = \begin{pmatrix} \beta_{S_1B_1,B_1J_1}\cdot\beta_{B_1J_1,J_1R_1} & \beta_{S_1B_1,B_1J_2}\cdot\beta_{B_1J_2,J_2R_2} \\ \beta_{S_2B_2,B_2J_1}\cdot\beta_{B_2J_1,J_1R_1} & \beta_{S_2B_2,B_2J_2}\cdot\beta_{B_2J_2,J_2R_2} \end{pmatrix}$$

Fig. 10. Comparison of our approach to [29] in the example of a 2-by2 topology. M_1, M_2, M_3, M_4 are the transfer matrices resulting from the four different types (types 1,2,3,4 in Fig. 3) of a 2-by-2, if intermediate nodes use coding coefficients β. The approach in [25], [29] tries to distinguish among these four M's in a single experiment. In contrast, we use β's either 0 or 1 and multiple experiments to choose an M.

TABLE III

Inference with Random Network Coding at intermediate nodes. POSSIBLE OBSERVATIONS FOR ALL FOUR TYPES OF 2-BY-2 TOPOLOGIES WHEN J_1 AND J_2 USE (1,1), (2,3) CODING COEFFICIENTS, RESPECTIVELY. (LOSSY OBSERVATIONS ARE OMITTED DUE TO LACK OF SPACE.)

Obs.	Type 1 (group (ii) obs.)		Type 2 (group (iii) obs.)		Type 3 (group (iii) obs.)		Type 4 (group (iii) obs.)	
#	$\mathbf{R_1}$	$\mathbf{R_2}$	$\mathbf{R_1}$	$\mathbf{R_2}$	$\mathbf{R_1}$	$\mathbf{R_2}$	$\mathbf{R_1}$	$\mathbf{R_2}$
1	x_1+x_2	x_1+x_2	x_1+x_2	$2(x_1+x_2)$	$2(x_1+x_2)$	x_1+x_2	x_1+x_2	$2x_1+3x_2$
2	x_1	x_1	x_1	$2x_1$	$2x_1$	x_1	x_1	$2x_1$
3	x_2	x_2	x_2	$3x_2$	$3x_2$	x_2	x_2	$3x_2$
4			x_1+x_2	$2(x_1+x_2)+3x_2$	$2(x_1+x_2)+3x_2$	x_1+x_2	x_1	$2x_1+3x_2$
5			x_1	$2x_1+3x_2$	$2x_1+3x_2$	x_1	x_1+x_2	$2x_1$
6			x_1	$3x_2$	$3x_2$	x_1	x_1	$3x_2$
7			x_1+x_2	$3x_2$	$3x_2$	x_1+x_2	x_2	$2x_1$
8							x_1+x_2	$3x_2$
9							x_2	$2x_1+3x_2$

of columns, which can be treated as an independent 2-by-2 transfer matrix (like what we described above). After all $\binom{N}{2}$ 2-by-2's are identified, we use the merging algorithm that in each step (*i.e.,* every time one J is localized) evolves F, the adjacency matrix of the line graph[11], as follows.

By localizing each joining point, more edges (as rows and columns) are added to F. In Section V-C1, we have part of F from the assumption about the knowledge of the S_1 1-by-N tree topology. As we localize each joining point to a single logical link between two branching points, we break the original edge in the S_1 topology into two parts, separated by the identified joining point. We also add the corresponding edges for the paths from S_2 to the joining point, and then to the receivers. This is shown in the following example. Note that in Section V-C2, we do not have any part of F in advance and we build F from scratch by finding all the edges from our 2-by-2 information.

Example 7: Consider the simple 2-by-3 network in Fig. 11. We start from the S_1 topology represented by the following F, corresponding to the edges $S_1B_{1,2}, B_{1,2}B_{2,3}, B_{1,2}R_1, B_{2,3}R_2, B_{2,3}R_3$:

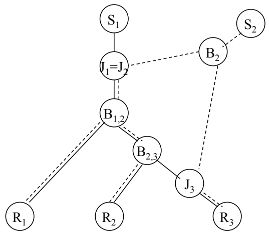


Fig. 11. A simple 2-by-3 topology. F evolves using the 2-by-2 information.

$$F = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Here the required 2-by-2 types are $S_1, S_2 - R_1, R_2$ and $S_1, S_2 - R_2, R_3$. Since $S_1, S_2 - R_1, R_2$ is of type 1, we identify $J_1 = J_2$ on $S_1B_{1,2}$. Thus, we divide $S_1B_{1,2}$ into S_1J_1 and $J_1B_{1,2}$. Since $S_1, S_2 - R_2, R_3$ is of type 2, we identify J_3 on $B_{2,3}R_3$. Thus, we divide $B_{2,3}R_3$ into $B_{2,3}J_3$ and J_3R_3. If all three 2-by-2's were of type 1, then S_2 would share the branching point $B_{1,2}$ with S_1. But now we need to consider B_2 for S_2 and add all the branches to the joining points $J_1 = J_2$ and J_3 from B_2. Therefore, we add S_2B_2, B_2J_1, and B_2J_3

[11] Under random linear network coding, the transfer matrix M can be written as $M = A(I-F)^{-1}B^T$ [44]. A and B^T represent the linear mixing of the input and output random processes, respectively. Therefore, they do not substantially contribute to M. F is the $|E|.|E|$ adjacency matrix of the line graph, where E denotes the number of edges in the graph, which completely describes the topology.

Algorithm 8 Algorithm 7 in terms of evolving the adjacency matrix of the line graph, F.

```
for each edge $e_k$ breaking into two edges do
    add one row $e_{k+1} = [0, 0, ..., 0]$ to $F_{i-1}$, which is of length $size(F_{i-1}) + 1$.
    add one column $e^T_{k+1}$.
    transfer all 1 elements from row $e_k$ to row $e_{k+1}$ in the new matrix $F_i$, and only make the $(e_k, e^T_{k+1})$ element 1.
end for
```

to F; where B_2J_1 meets with S_1's routes to R_1 and R_2 at J_1, and B_2J_3 meets with S_1's path to R_3 at J_3. Thus, we get the final matrix F, corresponding to the topology in Fig. 11, with edges S_1J_1, S_2B_2, B_2J_1, B_2J_3, $J_1B_{1,2}$, $B_{1,2}B_{2,3}$, $B_{2,3}J_3$, $B_{1,2}R_1$, $B_{2,3}R_2$, and J_3R_3:

$$F = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

■

In general, any 2-by-2 of type 3 or 4, limits the upper bound on the joining point location for the receiver under consideration; thus, breaks the current edge e_k into two consecutive edges. Assume that we have F_{i-1} at the current step, the 2-by-2 type we have considered breaks edge e_k into two edges, and we want to find F_i from F_{i-1}. We need to do the following steps: (i) add one row $e_{k+1} = [0, 0, ..., 0]$ to F_{i-1}, which is of length $size(F_{i-1}) + 1$; (ii) also add one column e^T_{k+1}; (iii) transfer all 1 elements from row e_k to row e_{k+1} in the new matrix F_i, and only make the (e_k, e^T_{k+1}) element 1. These steps are summarized in Alg. 8.

C. *Extension to Passive Tomography with RNC*

We now discuss how our approach can potentially be extended to be implemented as passive, when random network coding (RNC) is used in the middle. The intuition is that the same algorithms for topology inference should apply if we ensure that the RNC coefficients satisfy necessary conditions for inference.

Assume that in each experiment, the intermediate nodes perform random linear network coding operations instead of the simple additions assumed so far. In this case, Algorithm 6 still works if we assign coding coefficients to the joining points in a partial order, so as to ensure that the minimum coding coefficient of a joining point is always greater than or equal to the maximum coding coefficient of its ancestor joining point. Under this condition, we can prove that the same rationale as in Section V-B2 still holds, *i.e.*, type 1 always results in similar observations; in type 2, we have $c_{12} - c_{22} \leq 0$; in type 3, we have $c_{12} - c_{22} \geq 0$; and in type 4, both cases are possible.

Proof: As we can see in Fig. 3 (b) and (c), after the equal chance of the two packets x_1, x_2 to meet at J_1, x_2 has an extra chance of meeting with the resulting packet from J_1 at J_2, which would result in $c_{12} - c_{22} \leq 0$ in type 2, or $c_{12} - c_{22} \geq 0$ in type 3. However, our assumption about the partially ordered coefficients is still necessary; *e.g.*, assume that the two packets do not arrive within the same W at J_2 in type 2: if x_2 arrives within an earlier W at J_2 than the resulting packet from J_1, then it must get multiplied by a larger coefficient at J_2, rather than the one it already carries from J_1, so that $c_{12} - c_{22} \leq 0$ still holds at the receives. Therefore, the condition on the partial order of coding coefficients is required for our algorithms to be applicable to types 2 and 3. Type 4 is simply distinguishable from type 1 by getting different observations due to two different sets of coding coefficients at J_1, J_2; rather than the same coding coefficients from the single joining point $J_1 = J_2$ in type 1. ■

Code design to jointly meet both random network coding goals (large enough field for independent linear equations) and tomographic goals (the aforementioned condition) is part of future work. If such a code design is possible, Alg. 6 can be directly applied to the case of random network coding. For example in Fig. 3, let J_1 use (1,1) and J_2 use (2,3) to combine the incoming packets. Table III shows all possible observations. We can see that c_{12} is greater than, equal to, or smaller than c_{22}, depending on the number of joining points a probe packet meets on its way towards the two receivers. This is the same rationale and pattern as in Table II.

D. *Comparison to* `traceroute`*-like approaches*

In practice, the dominant approach to Internet mapping is based on `traceroute` [30]–[39]. It uses `traceroute`'s sent between selected nodes and collects the ids of the nodes along the paths traversed. It faces the challenges of (i) resolving anonymous routers and router aliases and (ii) causing congestion close to the monitoring points [37].

Similarly to `traceroute`, we also use active end-to-end probes and we require some minimal co-operation from internal nodes (simple additions in our case vs. traceroute-specific responses). However, unlike `traceroute`, we do not ask intermediate nodes to reveal their node id, which has the advantage of preserving the anonymity of intermediate nodes. A design difference was also noted in Section V-B4: we infer 2-by-2 components, instead of 1-by-1's (path) for `traceroute`.

In terms of measurement bandwidth, our approach uses exactly one probe per link per experiment, which is the minimum possible. This is thanks to network coding that combines multiple incoming packets into one, and thanks to multicast that replicates a single incoming packet into many outgoings, thus eliminating overlap. For example, our approach reduces the *number of measured paths* in a 2-by-N topology by a factor of two, compared to `traceroute`; *i.e.*, we require $O(N)$ instead of $O(2N)$ measurements, since each coded packet observes two paths.

In order to provide a quantitative comparison of our approach vs. traceroute, we compare the *average number of packets per link*, in type 1 and type 4 topologies. Standard traceroute sends three probe packets for each hop count, in each source-destination pair. Assuming that all nodes are responsive, traceroute results in 14.4 packets/link if the underlying topology is of type 1, and 9 packets/link if the underlying topology is of type 4. In contrast, in our approach, each link is traversed by a probe packet exactly once in each experiment. It is well-known that traceroute results in increased overhead on links close to sources and receivers.[12] Network coding allows flows to share bottlenecks without competition. We note however, that although we are efficient in a single experiment (one probe per link) and we use small probes, we repeat our experiments for up to $countMax$ times; by adjusting $countMax$, we trade-off accuracy for the load.

VIII. Conclusion

In this paper, we design active probing schemes that exploit simple operations at the intermediate nodes to accurately infer the network topology, based on end-to-end observations. We design algorithms for trees and general topologies, and simulate them in representative examples. Our main contribution is that we show how to exploit the fundamental connection between network coding and topology and thus adding one new building block in the space of available options for topology inference. The application context depends on the capabilities and constraints of the network. We expect the techniques developed in this paper to be most useful in networks that are already, or can be easily, equipped with network coding capabilities, such as overlay or wireless mesh networks.

References

[1] C. Fragouli, A. Markopoulou, and S. Diggavi, "Topology inference using network coding," in *the Allerton Conference*, Monticello, IL, Sept. 2006.

[2] M. Rabbat, M. Coates, and R. Nowak, "Multiple source internet tomography," *IEEE JSAC*, vol. 24, no. 12, pp. 2221–2234, Dec. 2006.

[3] M. Rabbat, R. Nowak, and M. Coates, "Multiple source multiple destination network tomography," in *IEEE INFOCOM*, Hong Kong, Mar. 2004.

[4] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: Recent developments," *Statistical Science*, vol. 19, no. 3, pp. 499–517, 2004.

[5] S. Ratnasamy and S. McCanne, "Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements," in *IEEE INFOCOM*, New York, NY, March 1999, pp. 353–360.

[6] N. G. Duffield, J. Horowitz, F. L. Presti, and D. Towsley, "Multicast topology inference from measured end-to-end loss," *IEEE Transactions on Information Theory*, vol. 48, pp. 26–45, Jan. 2002.

[7] N. G. Duffield and F. L. Presti, "Network tomography from measured end- to-end delay covariance," *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 978–992, Dec. 2004.

[8] M. Coates, R. Castro, M. Gadhiok, R. King, Y. Tsang, and R. Nowak, "Maximum likelihood network topology identification from edge-based unicast measurements," in *ACM Sigmetrics*, CA, USA, June 2002.

[9] R. Caceres, N. G. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network-internal loss characteristics," *IEEE Transactions on Information Theory*, vol. 45, pp. 2462–2480, Nov. 1999.

[10] A. Bestavros, J. Byers, and K. Harfoush, "Inference and labeling of metric-induced network topologies," in *IEEE INFOCOM*, NY, June 2002.

[11] ——, "Inference and labeling of metric-induced network topologies," *IEEE Transactions on Parallel Distributed Systems*, vol. 16, no. 10, pp. 1053–1065, October 2005.

[12] N. Duffield, J. Horowitz, and F. L. Presti, "Adaptive multicast topology inference," in *IEEE INFOCOM*, Anchorage, AK, April 2001.

[13] M.-F. Shih and A. Hero, "Network topology discovery using finite mixture models," in *IEEE ICASSP*, Montreal, CA, May 2004, pp. 433–436.

[14] N. Duffield, J. Horowitz, F. L. Presti, and D. Towsley, "Multicast topology inference from end-to-end measurements," in *ITC Seminar on IP Traffic, Measurement, and Modeling*, Monterey, CA, Sept. 2000.

[15] N. Duffield, F. L. Presti, V. Paxson, and D. Towsley, "Inferring link loss using striped unicast probes," in *IEEE INFOCOM*, Alaska, April 2001.

[16] R. Caceres, N. Duffield, J. Horowitz, F. L. Presti, and D. Towsley, "Loss based inference of multicast network topology," in *the 38th IEEE Conference on Decision and Control*, Phoenix, AZ, Dec. 1999.

[17] B. Eriksson, G. Dasarathy, P. Barford, and R. Nowak, "Toward the practical use of network tomography for internet topology discovery," in *IEEE INFOCOM*, San Diego, CA, March 2010.

[18] Y. Tsang, M. Yildiz, P. Barford, and R. Nowak, "Network radar: tomography from round trip time measurements," in *ACM IMC*, Taormina, Sicily, Italy, Oct. 2004, pp. 175–180.

[19] C. Fragouli, J. Widmer, and J. Y. LeBoudec, "Network coding: an instant primer," *ACM SIGCOM CCR*, vol. 36, no. 1, pp. 63–68, January 2006.

[20] C. Fragouli and E. Soljanin, "Monograph on Network Coding: Fundamentals and Applications," in *Foundations and Trends in Networking*. NOW Publishers, 2007, vol. 2.

[21] M. Gjoka, C. Fragouli, P. Sattari, and A. Markopoulou, "Loss tomography in general topologies with network coding," in *IEEE GLOBECOM*, Washington DC, Nov. 2007.

[22] A. Markopoulou, C. Fragouli, and M. Gjoka, "A network coding approach to loss tomography," http://arxiv.org/abs/1005.4769, 2010.

[23] P. Sattari, A. Markopoulou, and C. Fragouli, "Multiple source multiple destination topology inference using network coding," in *NetCod Workshop*, EPFL, Switzerland, June 2009.

[24] T. Ho, B. Leong, Y.-H. Chang, Y. Wen, and R. Koetter, "Network monitoring in multicast networks using network coding," in *IEEE ISIT*, Adelaide, Australia, Sept. 2005, pp. 1977–1981.

[25] G. Sharma, S. Jaggi, and B. Dey, "Network tomography via network coding," in *ITA Workshop*, San Diego, CA, Feb. 2008.

[26] M. Jafarisiavoshani, C. Fragouli, S. Diggavi, and C. Gkantsidis, "Bottleneck discovery and overlay management in network coded peer-to-peer systems," in *ACM SIGCOMM INM Workshop*, Kyoto, Japan, August 2007.

[27] M. Jafarisiavoshani, C. Fragouli, and S. Diggavi, "Subspace properties of randomized network coding," in *ITW*, Bergen, Norway, July 2007.

[28] H. Yao, S. Jaggi, and M. Chen, "Network coding tomography for network failures," in *IEEE INFOCOM*, San Diego, CA, March 2010.

[29] ——, "Passive network tomography for erroneous networks: A network coding approach," http://arxiv.org/abs/0908.0711, 2010.

[30] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in *IEEE INFOCOM*, Israel, March 2000, pp. 1371–1380.

[31] B. Cheswick, H. Burch, and S. Branigan, "Mapping and visualizing the internet," in *USENIX ATC*, San Diego, CA, June 2000, pp. 1–12.

[32] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," in *ACM Sigcomm*, Pittsburgh, PA, August 2002, pp. 133–146.

[33] B. Yao, R. Viswanathan, F. Chang, and D. Waddington, "Topology inference in the presence of anonymous routers," in *IEEE INFOCOM*, San Francisco, CA, March 2003, pp. 353–363.

[34] A. Clauset and C. Moore, "Why mapping the internet is hard," arXiv:cond-mat/0407339, 2004.

[35] L. Dall'Asta, I. Alvarez-Hamelin, A. Barrat, A. Vazquez, and A. Vespignani, "A statistical approach to the traceroute-like exploration of networks: Theory and simulations," *Lecture Notes in CS*, vol. 3405, p. 140, 2005.

[36] B. Huffaker, D. Plummer, D. Moore, and k. claffy, "Topology discovery by active probing," in *Symposium on Applications and the Internet*, Nara, Japan, January 2002, p. 90.

[12] An improved algorithm, called "Doubletree", was proposed in [37] to reduce redundancy of classic traceroute, while maintaining nearly the same level of node and link coverage, by having the sources share information regarding the paths they have explored. This approach further reduces the average overhead to 10.8 packets/link for type 1, and to 6.75 packets/link for type 4.

[37] B. Donnet, P. Raoult, T. Friedman, M. Crovella, T. Friedman, and R. Teixeira, “Deployment of an algorithm for large-scale topology discovery,” *JSAC*, vol. 24, no. 12, pp. 2210–2220, 2006.

[38] F. Viger, B. Augustin, X. Cuvellier, C. Magnien, T. F. M. Latapy, and R. Teixeira, “Detection, understanding, and prevention of traceroute measurement artifacts,” *Computer Networks*, vol. 52, no. 5, pp. 998–1018, April 2008.

[39] Y. Shavitt and E. Shir, “Dimes: Let the internet measure itself,” *ACM SIGCOMM CCR*, vol. 35, no. 5, pp. 71–74, October 2005.

[40] M. Coates, M. Rabbat, and R. Nowak, “Merging logical topologies using end-to-end measurements,” in *ACM IMC*, Miami, FL, Oct. 2003.

[41] “The abilene network,” http://noc.net.internet2.edu/.

[42] “Leda software,” http://www.algorithmic-solutions.com/.

[43] “Brite topology generator,” http://www.cs.bu.edu/brite/.

[44] R. Koetter and M. Medard, “An algebraic approach to network coding,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, October 2003.

Pegah Sattari (M’08) received the B.S. degree in Electrical Engineering from Sharif University of Technology, Tehran, Iran, in 2006, and the M.S. degree in Electrical and Computer Engineering from the University of California, Irvine, in 2007. She is currently a Ph.D. candidate in the EECS Department at UC Irvine. Her research interests include network measurements, network coding, and its applications to inference problems.

Christina Fragouli is an Assistant Professor in the School of Computer and Communication Sciences, EPFL, Switzerland. She received the B.S. degree in Electrical Engineering from the National Technical University of Athens, Athens, Greece, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of California, Los Angeles. She has worked at the Information Sciences Center, AT&T Labs, Florham Park New Jersey, and the National University of Athens. She also visited Bell Laboratories, Murray Hill, NJ, and DIMACS, Rutgers University. From 2006 to 2007, she was an FNS Assistant Professor in the School of Computer and Communication Sciences, EPFL, Switzerland. She served as an editor for IEEE Communications Letters, and is currently serving as an editor for IEEE Transactions on Information Theory, IEEE Transactions on Communications and for Elsevier Computer Communications. Her research interests are in network coding, network information flow theory and algorithms, and connections between communications and computer science. She received the Fulbright Fellowship for her graduate studies, the Outstanding Ph.D. Student Award 2000-2001, UCLA, Electrical Engineering Department, the Zonta award 2008 in Switzerland, and the Young Investigator ERC grant in 2009.

Athina Markopoulou (SM ’98, M’02) is an assistant professor in the EECS Dept. at the University of California, Irvine. She received the Diploma degree in Electrical and Computer Engineering from the National Technical University of Athens, Greece, in 1996, and the M.S. and Ph.D. degrees, both in Electrical Engineering, from Stanford University in 1998 and 2003, respectively. She has been a postdoctoral fellow at Sprint Labs (2003) and at Stanford University (2004-2005), and a member of the technical staff at Arastra Inc. (2005). Her research interests include network coding, network measurements and security, media streaming and online social networks. She received the NSF CAREER award in 2008.